Section 4

SCSI Bus Interface Circuits

Section 4 Contents

The DP8490 and DP5380 Comparison Guide

National Semiconductor
Application Brief 39
Desmond Young

OVERVIEW

National Semiconductor has released two products to support the Small Computer System Interface (SCSI). These two products offer significant features over existing asynchronous SCSI devices such as:

- Lower power (25 mW vs 700 mW)
- Lower cost
- Higher speed (up to 4 Mbytes/sec)
- More functionality
- Compatibility (with existing NCR5380)

FEATURES OF NATIONAL's DP8490 & DP5380

Low Power

The DP8490 Enhanced Asynchronous SCSI Interface (EASI) and the DP5380 Asynchronous SCSI Interface (ASI) are CMOS devices. The current drawn by the NCR5380 is 140 mA while both the EASI and the ASI draw 4 mA max—*and* the EASI/ASI are specified with SCSI pins terminated with SCSI spec values!

Low Cost

National's marketing strategy will ensure the EASI and ASI are the most competitively priced parts on the market.

High Speed

The high speed CMOS process used for the EASI and ASI means that DMA rates of up to 4 Mbytes per second are achievable. Currently there are no low cost high speed 8-bit DMA controllers that will provide the necessary speed. A common solution is to provide the simple DMA functionality as part of ASIC circuitry normally found in each application. National has published an application note on how to build a 4 Mbytes/sec asynchronous SCSI interface using the EASI/ASI devices to assist designers.

The user should be aware that with this increase in speed comes the need and care on the user's part in board design and layout. The EASI and ASI have reduced: read access times from 140 ns to 50 ns, write data hold times from 30 ns to 10 ns, and all other parameters correspondingly. This means the designer should ensure he has correctly generated chip strobes. In particular pay attention to the generation of DACK, and the period of hand-over of the bus between the DMA and CPU.

Improved Functionality

The EASI provides new and enhanced functions to ease the firmware overhead. The major features are:

new interrupts
new interrupt status and mask structure
extended arbitration
loopback and other testing facilities
microprocessor data bus parity
improved SCSI and DMA timing

New Interrupts

The EASI adds interrupts for: arbitration complete, any phase mismatch, true end of DMA and microprocessor parity error. The arbitration complete interrupt allows the firmware to program the EASI to arbitrate for the bus and interrupt when done. True End of DMA interrupt overcomes the fault in the DP5380 when using it to *send* data on SCSI.

New Interrupt Structure

The DP5380 cannot be fully interrupt driven (e.g., no arbitration complete interrupt)—the DP8490 can be! In the DP5380 the firmware must read 2 registers and then process the results to determine the cause of an interrupt. Not all interrupt conditions have a status bit in the DP5380. The DP8490 provides an interrupt status register which gathers all status into one register, with a dedicated status bit for each interrupt. An interrupt mask register provides individual masking control of each interrupt. Finally, the design of the interrupt logic ensures interrupts will not be lost (they can be on the DP5380).

Extended Arbitration

The DP5380 device must be polled by the μP during arbitration. Since this process does not start until the bus is free, the μP may be polling for many milliseconds. The DP8490 provides an interrupt on arbitration complete to allow the μP to handle other tasks such as cacheing data, overlapped seeks etc. The DP8490 also implements the 2.2 μs SCSI arbitration delay before interrupting the μP.

Loopback Testing

The DP8490 provides a loopback test mode. Each SCSI pin driver is disabled and looped back internally. The firmware may exercise the EASI to verify correct part operation. SCSI signal driving and monitoring, interrupts and DMA can all be tested. The EASI also provides programmable parity polarity (EVEN/ODD) for both μP and SCSI buses. These bits may be used to verify parity circuitry on all boards in a SCSI system.

Microprocessor Parity

The PCC-packaged DP8490 provides an extra pin for μP data bus parity. This enables checked data to be maintained throughout a system—even including controller buffer memory.

Improved Timing

The DP5380 has some timing anomalies just like the NCR5380 on the SCSI and μP interfaces. The DP8490 fixes these.

COMPATIBILITY

The DP5380 is *completely* compatible with existing NCR5380 type devices—but offers higher speed and lower power.

The DP8490 is also completely compatible with existing NCR5380 type devices except for one register bit. An unused "test-mode" bit in the NCR5380 is replaced by an "Enhanced Mode" bit in the DP8490. Until this bit is set the DP8490 functions as a DP5380. Once set the enhanced features of the DP8490 are accessible. Since the DP8490 powers up in DP5380 mode *all DP5380 sockets should be able to use a DP8490!*

For more information see Section 8 of the DP8490 datasheet.

National Semiconductor

DP8490 Enhanced Asynchronous SCSI Interface (EASI)

General Description

The DP8490 EASI is a CMOS device designed to provide a low cost, high performance Small Computer Systems Interface. It complies with the ANS X3.131-1986 SCSI standard as defined by the ANSI X3T9.2 committee. It can act as both INITIATOR and TARGET, making it suitable for any application. The EASI supports selection, reselection, arbitration and all other bus phases. High-current open-drain drivers on chip reduce application chip count by interfacing direct to the SCSI bus. An on-chip oscillator provides all timing delays.

The DP8490 is pin and program compatible with the NMOS NCR5380 and CMOS DP5380 devices. NCR5380, DP5380 or AM5380 applications should be able to use it with no changes to hardware or software. The DP8490 includes new features which make this part more attractive for new designs and performance upgrades. These new features include μP data bus parity, programmable parity for both SCSI and μP busses, loopback test mode, improved arbitration support, faster timing and extended interrupt control logic. The DP8490 is available in a 40-pin DIP or a 44-pin PCC.

The EASI is intended to be used in a microprocessor based application, and achieves maximum performance with a DMA controller. The device is controlled by reading and writing several internal registers. A standard non-multiplexed address and data bus easily fits any μP environment.

Data transfers can be performed by programmed-I/O, pseudo-DMA or via a DMA controller. The EASI easily interfaces to a DMA controller using normal or Block Mode. The EASI can be used in either a polled or interrupt-driven environment. The EASI includes enhanced features for interrupt control.

Features

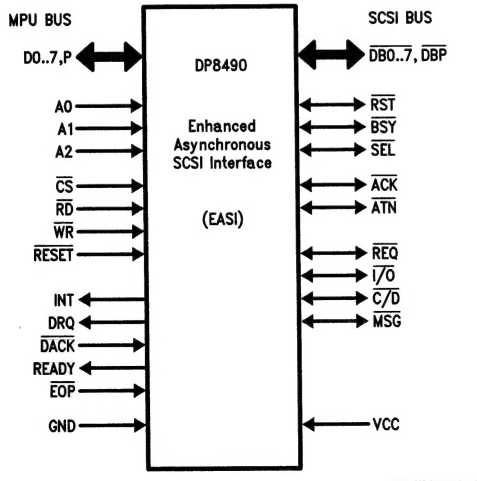
SCSI Interface

- Supports TARGET and INITIATOR roles
- Parity generation with optional checking
- Programmable parity polarity (ODD/EVEN)
- Arbitration support—can interrupt when done
- Direct control/monitoring of all SCSI signals
- High current outputs drive SCSI bus directly
- Faster and improved timing
- Very low SCSI bus loading

μP Interface

- Memory or I/O-mapped control transfers
- Programmed-I/O or DMA data transfers
- Normal or Block-mode DMA
- Fast DMA handshake timing
- Individually maskable interrupts
- Active interrupts identified in one register
- Optional data bus parity generation/checking
- Programmable parity polarity (ODD/EVEN)
- Loopback test mode

Connection Diagram



TL/F/9387–1

Table of Contents

1.0 Functional Description

1.1 OVERVIEW

The EASI is designed to be used as a peripheral device in a μP-based application and appears as a number of read/write registers. Write registers are programmed to select desired functions. Status registers provide indication of operating conditions. In an application extensive use of interrupts is desirable. The EASI incorporates an improved interrupt structure which enables fully interrupt-driven operation. In the enhanced mode interrupts can be individually masked or enabled, and a status register identifies all active interrupt requests.

For best performance a DMA controller can be easily interfaced directly to the EASI. The EASI provides request/acknowledge and wait-state signals for the DMA interface.

The SCSI bus is easily controlled via the EASI registers. Any bus signal may be asserted or deasserted via a bit in the appropriate register, and the state of every signal is available by reading registers. This direct control over SCSI signals allows the user to implement all or part of the protocol in firmware. The EASI provides hardware support for much of the protocol, and all speed-critical steps are handled by the EASI.

The EASI provides the following SCSI support:

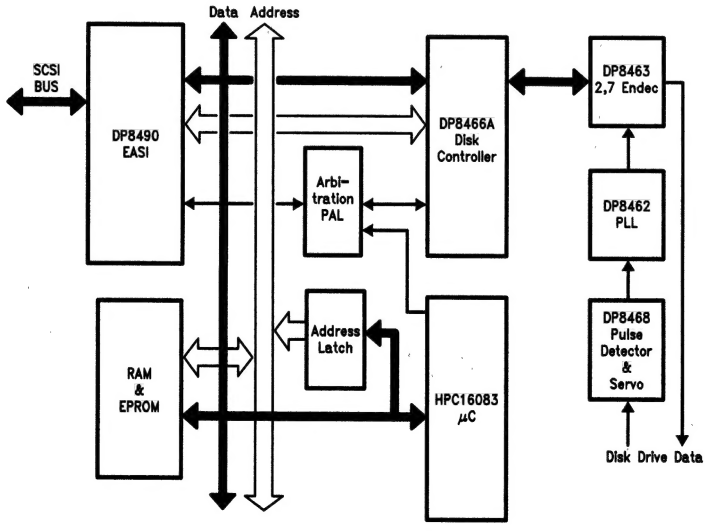
- Programmed-I/O transfers for all eight information transfer types, with or without parity.
- Data transfers via DMA, in either block or non-block mode. The DMA interface supports most devices.
- Individual setting/resetting and monitoring of every SCSI bus signal.
- Automatic release of the bus for BSY loss from a TARGET, SCSI RST, and lost arbitration.
- Automatic bus arbitration with an optional interrupt upon completion—the μP has only to check for highest priority. The 2.2 μs arbitration delay can be optionally performed by the EASI.
- Selection or Reselection of any bus device. The EASI will respond to both Selection and Reselection.
- Optional automatic monitoring of the $\overline{\text{BSY}}$ signal from a TARGET with an interrupt after releasing control of the bus.
- Optional parity polarity selection. Default after reset is ODD, but EVEN generation and checking can be programmed for diagnostic purposes and to determine whether a device supports parity when first making a connection.

Figure 1 shows an EASI in a typical application, a low cost embedded SCSI disk controller. In this application the 8051 single-chip μP acts as the controller and the dual DMA channels in the DP8475 allow one for the disk data and the other for SCSI data. The PAL provides chip selection as well as determining who has control of the bus. The advantage of using a μP with on-board ROM is that there is more free time on the external bus.

1.2 μP INTERFACE

Figure 2 shows a block diagram of the EASI. Key blocks within the EASI are Read/Write registers with associated decode and control logic, interrupt and DMA logic, SCSI bus arbitration logic, SCSI drivers/receivers with parity and the SCSI data input and output registers. The EASI has three interfaces, one to SCSI, one to a DMA controller and the third to a μP. The internal registers control all operation of the EASI.

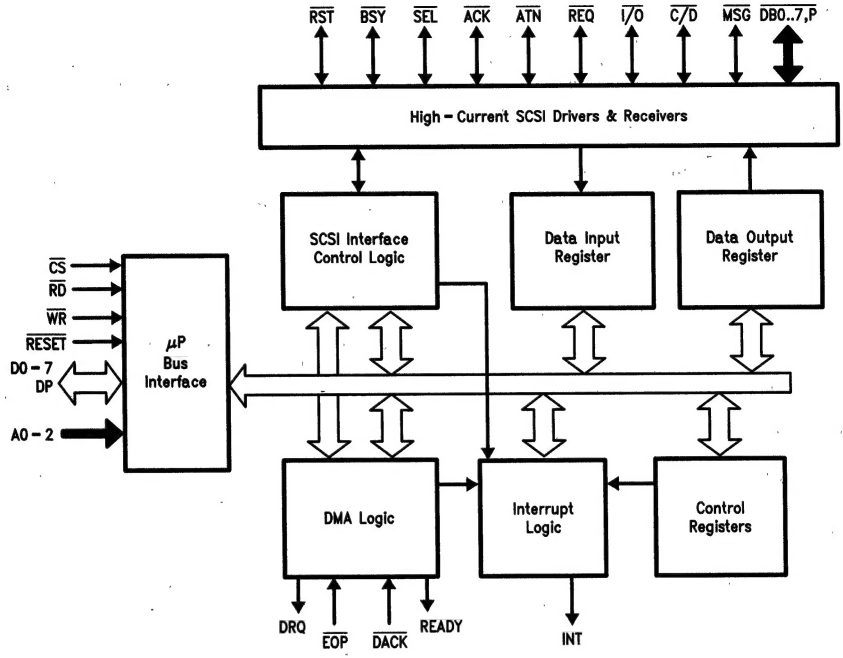
The μP interface consists of non-multiplexed address and data busses with associated control signals. The data bus can be programmed to use either ODD or EVEN parity. Address decode logic selects a register for reading or writing. The address lines A0–A2 select the register for μP accesses while for DMA accesses the address lines are ignored. The decode logic also selects different registers or functions to be mapped into address 7, according to the programmed mode (see Section 8).



TL/F/9387–2

FIGURE 1. EASI Application

1.0 Functional Description (Continued)



TL/F/9387–3

FIGURE 2. EASI Block Diagram

The register bank consists of twelve registers mapped into an address space of eight locations. Upon an external chip reset the registers are cleared (all zeroes)—the same as the NCR5380. Once the ENHANCED MODE bit in the INITIATOR COMMAND REGISTER is set, three new registers can be accessed to utilize the extra features of the DP8490 EASI.

1.3 DMA INTERFACE

The DMA logic interfaces to single-cycle, block mode, flow-through or fly-by controllers. Single byte transfers are accomplished via the DRQ/$\overline{\text{DACK}}$ handshake signals. Block mode transfers use the READY output to control the speed (insert wait-states). An End Of Process ($\overline{\text{EOP}}$) input from the DMA controller signals the EASI to halt DMA transfers. An interrupt can be generated for DMA completion or an error (see Section 5). All DMA data passes through the SCSI data input and output registers, automatically selected during DMA cycles.

1.4 SCSI INTERFACE

The EASI contains all logic required to interface directly to the SCSI bus. Direct control and monitoring of all SCSI signals is provided. The state of each SCSI signal may be determined by reading a register which continuously reflects the state of the bus. Each signal may be asserted by writing a ONE to the appropriate bit.

The EASI includes logic to automatically handle SCSI timing sequences too fast for µP control. In particular there is hardware support for DMA transfers, bus arbitration, selection/reselection, bus phase monitoring, $\overline{\text{BSY}}$ monitoring for bus disconnection, bus reset and parity generation and checking.

The EASI arbitration logic controls arbitration for use of the SCSI bus. The µP programs the SCSI device ID into the EASI, then sets the ARBITRATE bit. The EASI will interrupt the µP when one of three events occurs: arbitration is lost; arbitration has completed and the ID priorities need to be checked; or arbitration is complete and the 2.2 µs SCSI Arbitration delay has expired. Arbitration can be invoked with the enhanced feature of an interrupt on completion or the expiration of the SCSI Arbitration delay. These extra steps are programmed via the EXTRA MODE REGISTER (EMR). The INITIATOR COMMAND REGISTER (ICR) is read to determine whether arbitration has been won or lost.

The $\overline{\text{BSY}}$ signal is continuously monitored to detect bus disconnection and bus free phases. The EASI incorporates an on-board oscillator to determine Bus Settle, Bus Free and Arbitration Delays. The oscillator tolerance guarantees all timing to be within the SCSI specification.

The EASI incorporates high-current drivers and SCHMITT trigger receivers for interfacing directly to the SCSI bus. This feature reduces the chip count of any SCSI application. The driver/receivers also incorporate loopback logic which is enabled by an EMR bit. The Loopback mode enables testing of all EASI functions without interfering with the SCSI bus.

1.5 PARITY

The EASI provides for parity protection on both the µP and SCSI interfaces. Each data bus has eight data bits and one parity bit (only the PCC part provides µP parity, both the DIP and PCC provide SCSI parity). In each case the parity may

1.0 Functional Description (Continued)

be enabled via a register bit. A parity error can be programmed to cause an interrupt. Additionally the parity may be programmed to be either ODD or EVEN. This has a particular use on the SCSI interface where programming EVEN parity allows diagnostics, or determining whether a device supports parity. The inclusion of μP parity allows development of controllers that maintain data integrity right from the media to the host system.

1.6 INTERRUPTS

The EASI is intended to be used in an interrupt-driven environment. Each function can be programmed to cause an interrupt. In ENHANCED MODE two registers are used to control interrupts—the INT STATUS REGISTER (ISR) and the INT MASK REGISTER (IMR). Each interrupt can be masked from interrupting via the IMR. When an interrupt is recognized by the μP, reading the ISR will display all active interrupt sources. The ISR contents remain unchanged until an interrupt reset is programmed. A shadow register behind the ISR guarantees that interrupts occurring while others are serviced will not be lost.

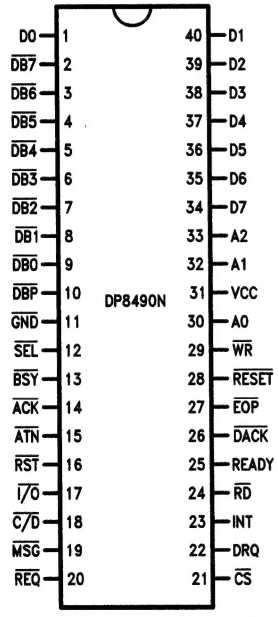
2.0 Pin Descriptions

Symbol	DIP	PCC	Type	Function
$\overline{CS}$	21	24	I	**Chip Select:** an active low enable for read or write operations, accessing the register selected by A0–A2.
A0–A2	30, 32 33	33, 36 37	I	**Address 0–2:** these three signals are used with $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ to address a register for read or write.
$\overline{RD}$	24	27	I	**Read:** an active low enable for reading an internal register selected by A0–A2 and enabled by $\overline{CS}$. It also selects the Input Data Register when used with $\overline{DACK}$.
$\overline{WR}$	29	32	I	**Write:** an active low enable for writing an internal register selected by A0–A2 and enabled by $\overline{CS}$. It also selects the Output Data Register when used with $\overline{DACK}$.
$\overline{RESET}$	28	31	I	**Reset:** an active low input with a Schmitt trigger. Clears all internal registers. (SCSI $\overline{RST}$ unaffected.)
D0–D7, P	1, 40–34	2, 44–38 1	I/O	**Data 0–7, P:** bidirectional TRI-STATE® signals connecting the active high μP data bus to the internal registers. The PCC part offers an optional parity on the μP data bus. If the parity option is not enabled the pin is TRI-STATE.
INT	23	26	O	**Interrupt:** an active high output to the μP when an error has occurred, an event requires service or has completed.
DRQ	22	25	O	**DMA Request:** an active high output asserted when the data register is ready to be read or written. DRQ occurs only if DMA mode is enabled. The signal is cleared by $\overline{DACK}$.
$\overline{DACK}$	26	29	I	**DMA Acknowledge:** an active low input that resets DRQ and addresses the data registers for input or output transfers. $\overline{DACK}$ is used instead of $\overline{CS}$ by the DMA controller.
READY	25	28	O	**Ready:** an active high output used to control the speed of block mode DMA transfers. Ready goes active when the chip is ready to send/receive data and remains inactive after the transfer until the byte is sent or until the DMA mode bit is reset.
$\overline{EOP}$	27	30	I	**End of Process:** an active low signal that terminates a block of DMA transfers. It should be asserted during the transfer of the last byte.
$\overline{DB0}$–$\overline{DB7}$ $\overline{DBP}$	9–2, 10	10–3, 11	I/O	**DB0–DB7, DBP:** SCSI data bus with parity. $\overline{DB7}$ is the MSB and is the highest priority during arbitration. Parity is default ODD but can be programmed EVEN. Parity is always generated and can be optionally checked. Parity is not valid during arbitration.
$\overline{RST}$	16	18	I/O	**Reset:** SCSI reset, monitored and can be set by EASI.
$\overline{BSY}$	13	15	I/O	**Busy:** indicates the SCSI bus is being used. Can be driven by TARGET or INITIATOR.

2.0 Pin Descriptions (Continued)

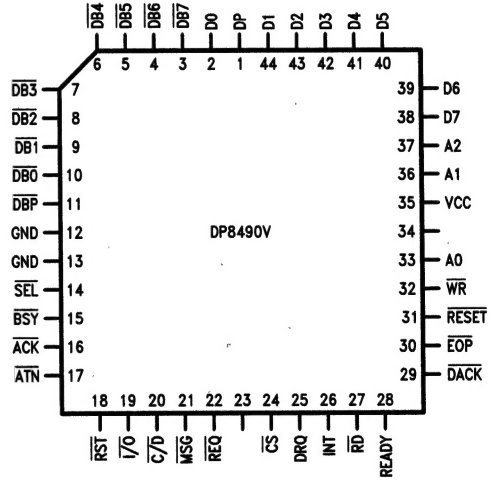
Symbol	DIP	PCC	Type	Function
$\overline{SEL}$	12	14	I/O	**Select:** used by the INITIATOR to select a TARGET or by the TARGET to reselect an INITIATOR.
$\overline{ACK}$	14	16	I/O	**Acknowledge:** driven by the INITIATOR and received by the TARGET as part of the $\overline{REQ}/\overline{ACK}$ handshake.
$\overline{ATN}$	15	17	I/O	**Attention:** driven by the INITIATOR to indicate an attention condition to the TARGET.
$\overline{REQ}$	20	22	I/O	**Request:** driven by the TARGET and received by the INITIATOR as part of the $\overline{REQ}/\overline{ACK}$ handshake.
$\overline{I/O}$	17	19	I/O	**Input/Output:** driven by the TARGET to control the direction of transfers on the SCSI bus. This signal also distinguishes between selection and reselection.
$\overline{C/D}$	18	20	I/O	**Command/Data:** driven by the TARGET to indicate whether command or data bytes are being transferred.
$\overline{MSG}$	19	21	I/O	**Message:** driven by the TARGET during message phase to identify message bytes on the bus.
V_{CC} GND	31 11	35 12, 13	—	**V_{CC}, GND:** $+5$ V_{DC} is required. Because of very large switching currents, good decoupling and power distribution is mandatory.

2.1 Connection Diagrams



TL/F/9387–4

Order Number DP8490N
See NS Package Number N40A



TL/F/9387–5

Order Number DP8490V
See NS Package Number V44A

3.0 Register Description

3.1 GENERAL

The DP8490 EASI is a register-based device with eight addressable locations used to access twelve registers. Some addresses have dual functions depending upon whether they are being read from or written to. Two basic operating "modes" result in differences in the registers accessed through address 7. Device operation is described in Section 4 but mode differences are highlighted in this section and Section 8.

The EASI operates in one of two modes—NORMAL (MODE N) and ENHANCED (MODE E). Switching between the modes is performed by setting or resetting bit 6 in the Initiator Command Register.

In MODE N, EASI registers appear the same as the DP5380. In MODE E, address 7 accesses enhanced logic features. To help identify these differences the register description is split into two subsections. The first describes the registers in MODE N. The second describes the register differences when in MODE E.

Figure 3.1 summarizes the register map in MODE N. Note that for registers reading or writing SCSI signals the SCSI name is used for each bit. Although the SCSI bus is active low, the registers invert the SCSI bus. This means an active SCSI signal is represented by a ONE in a register and an inactive signal by a ZERO.

Hex Adr	Register	Mnemonic	Bits	R/W
0	Output Data Register	ODR	8	WO
0	Current SCSI Data	CSD	8	RO
1	Initiator Command Register	ICR	8	RW
2	Mode Register 2	MR2	8	RW
3	Target Command Register	TCR	4	RW
4	Select Enable Register	SER	8	WO
4	Current SCSI Bus Status	CSB	8	RO
5	Bus and Status	BSR	8	RO
5	Start DMA Send	SDS	0	WO
6	Start DMA Target Receive	SDT	0	WO
6	Input Data Register	IDR	8	RO
7	Start DMA Initiator Receive	SDI	0	WO
7	Reset Parity/Interrupts	RPI	0	RO

FIGURE 3.1. Normal Mode Registers

3.2 NORMAL MODE REGISTERS

OUTPUT DATA REGISTER (ODR)

8 Bits HA 0 Write Only

This is a transparent latch used to send data to the SCSI bus. The register can be written by µP cycles or via DMA. DMA writes automatically select the ODR at Hex Address 0 (HA 0). This register is also written with the ID bits required during arbitration and selection/reselection phases. Data is latched at the end of the write cycle.

Bit 7 **Bit 0**

$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Output Data Register

CURRENT SCSI DATA (CSD)

8 Bits HA 0 Read Only

This register enables reading of the current SCSI data bus. If SCSI parity checking is enabled it will be checked at the beginning of the read cycle. The register is also used for µP accesses of SCSI data during programmed-I/O or ID checking during arbitration. Parity is not valid during arbitration. DMA transfers select the IDR (HA 6) instead of the CSD register.

Bit 7 **Bit 0**

$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Current SCSI Data

INITIATOR COMMAND REGISTER (ICR)

8 Bits HA 1 Read/Write

This register is used to control the INITIATOR and some other SCSI signals, and to monitor the progress of bus arbitration. Most of the SCSI signals may also be asserted in TARGET mode. Bits 5 to 0 are reset when $\overline{BSY}$ is lost (see MR2 description).

Bit 7 **Bit 0**

$\overline{RST}$	AIP/ MODE	LA/ DIFF	$\overline{ACK}$	$\overline{BSY}$	$\overline{SEL}$	$\overline{ATN}$	DBUS

Initiator Command Register

DBUS: Assert Data Bus Bit 0

0 Disable SCSI data bus driving.

1 Enable contents of Output Data Register onto the SCSI data bus. SCSI parity is also generated and driven on DBP.

This bit should be set when transferring data out of the EASI in either TARGET or INITIATOR mode, for both DMA or programmed-I/O. In INITIATOR mode the drivers are only enabled if: Mode Register 2 TARGET MODE bit is 0, and $\overline{I/O}$ is false, and $\overline{C/D}$, $\overline{I/O}$, $\overline{MSG}$ match the contents of the Target Command Register (phasematch is true). In TARGET mode only the MR2 bit needs to be set with this bit.

Reading the ICR reflects the state of this bit.

$\overline{ATN}$: Assert Attention Bit 1

0 Deassert $\overline{ATN}$

1 Assert SCSI $\overline{ATN}$ signal. The MR2 TARGET MODE bit must also be false to assert the signal.

Reading the ICR reflects the state of this bit.

$\overline{SEL}$: Assert Select Bit 2

0 Deassert $\overline{SEL}$

1 Assert SCSI $\overline{SEL}$ signal. Can be used in INITIATOR or TARGET mode.

Reading the ICR reflects the state of this bit.

$\overline{BSY}$: Assert Busy Bit 3

0 Deassert $\overline{BSY}$.

1 Assert SCSI $\overline{BSY}$ signal. Can be used in INITIATOR or TARGET mode.

Reading the ICR reflects the state of this bit.

3.0 Register Description (Continued)

$\overline{ACK}$: Assert Acknowledge Bit 4

0 Deassert $\overline{ACK}$.

1 Assert SCSI $\overline{ACK}$ signal. The MR2 TARGET MODE bit must also be false to assert the signal.

Reading the ICR reflects the state of this bit.

DIFF: Differential Enable Bit 5 Write

0 This bit must be reset to 0.

LA: Lost Arbitration Bit 5 Read

0 Normally reset to 0 to show arbitration not lost or not enabled.

1 Will be set when the EASI loses arbitration, i.e. when SEL is true during arbitration AND the Assert $\overline{SEL}$ bit of this register is false.

A 1 in this bit means the EASI has arbitrated for the bus, asserted $\overline{BSY}$ and its ID on the data bus and another device has asserted $\overline{SEL}$. The ARBITRATE bit in MR2 or the EMR must be set to enable arbitration.

MODE: Operating Mode Bit 6 Write

0 Normal Mode (MODE N) is selected.

1 Enhanced Mode (MODE E) is selected.

AIP: Arbitration In Progress Bit 6 Read

0 Normally 0 to show no arbitration in progress.

1 Set when the EASI has detected BUS FREE phase and asserted $\overline{BSY}$ and the Output Data Register contents onto the SCSI data bus. This bit remains set until arbitration is disabled.

$\overline{RST}$: Assert $\overline{RST}$ Bit 7

0 Deassert $\overline{RST}$.

1 Assert SCSI $\overline{RST}$ signal. $\overline{RST}$ is asserted as long as this bit is 1, or until a μP Reset ($\overline{RESET}$).

After this bit is set the INT pin goes active and internal registers are reset (except for the interrupt latch, MR2 TARGET MODE bit, and this bit). Reading the ICR reflects the state of this bit.

MODE REGISTER 2 (MR2)

8 Bits HA 2 Read/Write

This register is used to program basic operating conditions in the EASI. Operation as TARGET or INITIATOR, DMA mode and type as well as some interrupt controls are set via this register. This is a read/write register and when read the value reflects the state of each bit.

Bit 7 **Bit 0**

BLK	TARG	PCHK	PINT	EOP	BSY	DMA	ARB

Mode Register 2

ARB: Arbitrate Bit 0

0 Disable arbitration

1 Enable arbitration. The EASI will wait for a BUS FREE phase then arbitrate for the bus. Before setting this bit the Output Data Register should contain the SCSI device ID—a single bit set only. The status of the arbitration process is given in the AIP and LA bits (6,5) in the Initiator Command Register.

DMA: DMA Mode Bit 1

0 Disable DMA mode

1 Enable DMA operation. This bit should be set then one of address 5 to 7 written to start DMA. The TARGET MODE bit in the ICR and the phase lines in the TCR should have been set appropriately. The DBUS bit in the ICR must be set for DMA send operations. $\overline{BSY}$ must be active in order to set this bit. The phase lines must match the contents of the TCR during the actual transfers. In DMA mode EASI logic automatically controls the $\overline{REQ}$/$\overline{ACK}$ handshakes.

This bit should be reset by a μP write to stop any DMA transfer. An $\overline{EOP}$ signal will not reset this bit. During DMA, $\overline{CS}$ and $\overline{DACK}$ should not be active simultaneously.

This bit will be reset if $\overline{BSY}$ is lost during DMA mode.

BSY: Monitor Busy Bit 2

0 Disable $\overline{BSY}$ monitor.

1 Monitor SCSI $\overline{BSY}$ signal and interrupt when $\overline{BSY}$ goes inactive. When this bit goes active the lower 6 bits of the ICR are reset and all signals removed from the SCSI bus. This is used to check for valid TARGET connection.

EOP: Enable $\overline{EOP}$ Interrupt Bit 3

0 No interrupt for $\overline{EOP}$.

1 Interrupt after valid $\overline{EOP}$ condition.

PINT: Enable SCSI Parity Interrupt Bit 4

0 No Interrupt on SCSI parity error.

1 When SCSI parity is enabled via the PCHK bit, setting this bit enables an interrupt upon a SCSI parity error.

PCHK: Enable SCSI Parity Checking Bit 5

0 No SCSI parity checking.

1 Enable checking of SCSI parity during read operations. This applies to either programmed I/O or DMA mode.

TARG: Target Mode Bit 6

0 Initiator Mode

1 Target Mode

BLK: Block Mode DMA Bit 7

0 Non-block DMA

1 When set along with DMA bit (1), enables block mode DMA transfers. In block mode the READY line is used to handshake each byte with the DMA controller instead of the DRQ/$\overline{DACK}$ handshake used in non-block mode.

TARGET COMMAND REGISTER (TCR)

4 Bits HA 3 Read/Write

This register is used to control TARGET SCSI signals and to program the desired phase during INITIATOR mode. During DMA transfers the SCSI phase lines ($\overline{C/D}$, $\overline{MSG}$, $\overline{I/O}$) must match the contents of the TCR for transfers to occur. A phase mismatch halts DMA transfers and generates an interrupt.

Bit 7 **Bit 0**

X	X	X	X	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$

Target Command Register

3.0 Register Description (Continued)

This is a read/write register and the value read reflects the state of each bit, except bits 4–7 which always read 0.

$\overline{I/O}$: Assert $\overline{I/O}$ Bit 0

0 Deassert $\overline{I/O}$.

1 Assert SCSI $\overline{I/O}$ signal. The MR2 TARGET MODE bit must also be active.

$\overline{C/D}$: Assert $\overline{C/D}$ Bit 1

0 Deassert $\overline{C/D}$

1 Assert SCSI $\overline{C/D}$ signal. The MR2 TARGET MODE bit must also be active.

$\overline{MSG}$: Assert $\overline{MSG}$ Bit 2

0 Deassert $\overline{MSG}$.

1 Assert SCSI $\overline{MSG}$ signal. The MR2 TARGET MODE bit must also be active.

$\overline{REQ}$: Assert $\overline{REQ}$ Bit 3

0 Deassert $\overline{REQ}$.

1 Assert SCSI $\overline{REQ}$ signal. The MR2 TARGET MODE bit must also be active. This bit is used to handshake SCSI data via programmed-I/O.

SELECT ENABLE REGISTER (SER)

8 Bits HA 4 Write Only

This write-only register is used to program the SCSI device ID for the EASI to respond to during Selection or Reselection phases. Only one bit in the register should be set. When $\overline{SEL}$ is true, $\overline{BSY}$ false and the SER ID bit active an interrupt will occur.

This interrupt is reset or can be disabled by writing zero to this register. Parity will also be checked during Selection or Reselection if the PCHK bit in MR2 is set.

Bit 7 Bit 0

$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Select Enable Register

CURRENT SCSI BUS STATUS (CSB)

8 Bits HA 4 Read Only

This read-only register is used to monitor SCSI control signals and the SCSI parity bit. The SCSI lines are monitored during programmed-I/O transfers and after an interrupt in order to determine the cause. A bit is 1 if the corresponding SCSI signal is active.

Bit 7 Bit 0

$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$	$\overline{SEL}$	$\overline{DBP}$

Current SCSI Bus Status

BUS AND STATUS REGISTER (BSR)

8 Bits HA 5 Read Only

This read-only register is used to monitor SCSI signals not included in the CSB, and internal status bits. This register is read after an interrupt (in MODE N) to determine the cause of an interrupt. Bit 0 or 1 are set to 1 if the SCSI signal is active.

Bit 7 Bit 0

EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

Bus and Status Register

$\overline{ACK}$: Acknowledge Bit 0

This bit reflects the state of the SCSI $\overline{ACK}$ Signal.

$\overline{ATN}$: Attention Bit 1

This bit reflects the state of the SCSI $\overline{ATN}$ Signal.

BSY: Busy Error Bit 2

0 No error

1 The SCSI $\overline{BSY}$ signal has become inactive while the MR2 $\overline{BSY}$ (Monitor $\overline{BSY}$) bit is set. This will cause an interrupt, remove all EASI signals from the SCSI bus and reset the DMA MODE bit in MR2.

PHSM: Phase Match Bit 3

0 Phase Match. The SCSI $\overline{C/D}$, $\overline{I/O}$ and $\overline{MSG}$ phase lines are continuously compared with the corresponding bits in the TCR. The result of this comparison is reflected in this bit. This bit must be 1 (phase matches) for DMA transfers. A phase mismatch will stop DMA transfers and cause an interrupt.

I'.r: Interrupt Request Bit 4

0 No interrupt

1 Interrupt request active. Set when an enabled interrupt condition occurs. This bit reflects the state of the INT pin. INT may be reset by performing a Reset Parity/Interrupt (RPI) function.

SPER: SCSI Parity Error Bit 5

0 No SCSI parity error

1 SCSI parity error occurred. This bit remains set once an error occurs until the RPI function clears it. The PCHK bit in MR2 must be set for a parity error to be checked and registered.

DRQ: DMA Request Bit 6

0 No DMA request

1 DMA request active. This bit reflects the state of the DRQ pin. DRQ is reset by asserting $\overline{DACK}$ during a DMA cycle or by resetting the DMA bit in MR2. A Busy error will reset the MR2 DMA bit and thus will also clear DRQ. A phase mismatch will not reset DRQ.

EDMA: End of DMA Bit 7

0 Not end of DMA

1 Set when $\overline{DACK}$, $\overline{EOP}$ and either $\overline{RD}$ or $\overline{WR}$ are active simultaneously. Normally occurs when the last byte is transferred by the DMA. During DMA send operations the last byte transferred by the DMA may not have been transferred on SCSI so $\overline{REQ}$ and $\overline{ACK}$ should be monitored to verify when the last SCSI transfer is complete. This bit is reset when the MR2 DMA bit is reset.

Note: In MODE E the EASI presents a true EDMA bit in bit 7 of the TCR. This feature removes the need to poll the $\overline{REQ}$ and $\overline{ACK}$ signals.

START DMA SEND (SDS)

0 Bits HA 5 Write Only

This write-only register is used to start a DMA send operation. A write of don't care data should be the last thing done by the μP. The MR2 DMA, BLK and TARG bits must have been programmed previously.

Bit 7 Bit 0

X	X	X	X	X	X	X	X

Start DMA Send

3.0 Register Description (Continued)

START DMA TARGET RECEIVE

0 Bits HA 6 Write Only

This write-only register is used to start a DMA Target Receive operation. Same comments as SDS apply.

INPUT DATA REGISTER (IDR)

8 Bits HA 6 Read Only

This read-only register contains the SCSI data last latched during a DMA receive. Each byte from SCSI is latched into this register automatically by the EASI DMA logic. A DMA read ($\overline{DACK}$ and $\overline{RD}$) automatically selects this register. Programmed I/O SCSI data reads should use the CSD (HA 8).

START DMA INITIATOR RECEIVE (SDI)

0 Bits HA 7 Write Only

This write-only register is used to start DMA INITIATOR Receive Operation. Same comments as SDS apply. An alternative method of performing the SDI function is available in the Enhanced Mode Register.

RESET PARITY/INTERRUPT (RPI)

0 Bits HA 7 Read Only

This read-only register is used to reset the parity and interrupt latches. Reading this register resets the SCSI parity, μP parity, Busy Loss and Interrupt Request latches. It also resets the interrupt latches presented in the Interrupt Status Register (available in MODE E).

An alternative method of performing the Reset Parity Interrupt function is available in the Enhanced Mode Register. In MODE E writing a value of 01 to bits 2 and 1 of the EMR will reset the same bits as a read from HA 7 in MODE N. The EMR RPI will also reset enhanced logic that has bits set in the Interrupt Status Register.

3.3 ENHANCED MODE REGISTERS

Addresses 0 to 6 remain the same as MODE N except for bit 7 of TCR, as described below. Address 7 is the SDI and RPI functions in MODE N, but in MODE E it directly accesses the Enhanced Mode Register (EMR) and indirectly accesses the Interrupt Mask Register (IMR) and the Interrupt Status Register (ISR).

When bit 6 in the ICR (HA 1) is set, HA 7 accesses the read/write EMR. The SDI and RPI functions performed when writing/reading HA 7 in MODE N are disabled. To perform these functions the EMR is used instead.

Note that EMR functions are intended to be used in an interrupt-driven environment. Reading this register reflects the state of each bit.

TARGET COMMAND REGISTER (TCR)

5 Bits HA 3 Read/Write

This is the same as MODE N except for bit 7 which is described below. Note bits 4–6 always read 0.

Bit 7 **Bit 0**

(true) EDMA	X	X	X	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$

Target Command Register

EDMA: True End of DMA Bit 7

0 Not End of DMA

1 Set when the last byte of data has been transferred. This bit is not set until $\overline{REQ}$ and $\overline{ACK}$ both go inactive following the DMA cycle during which $\overline{EOP}$ was asserted. Note that unlike the BSR EDMA bit, this bit reflects the true completion of DMA transfers.

ENHANCED MODE REGISTER (EMR)

8 Bits HA 7 Read/Write

This register is accessed at HA 7 when the ICR MODE bit (6) is set. The register controls operation of enhanced logic and timing. Normally the application will leave the EASI permanently in MODE N or MODE E.

Bit 7 **Bit 0**

APHS	MPEN	MPOL	SPOL	LOOP	EFN1	EFN0	ARB

Enhanced Mode Register

ARB: Extended Arbitration Bit 0

0 Disable extended arbitration

1 Enable extended arbitration. This is an alternative bit to the MR2 ARB function. The EASI waits for a BUS FREE phase then arbitrates for the bus, asserting the contents of the ODR onto the SCSI data bus and asserting $\overline{BSY}$. The EASI will then wait the 2.2 μs SCSI Arbitration Delay, and set the Arbitration Complete bit in the ISR and cause an interrupt. As for the MR2 ARB function, the ICR LA and AIP bit can be examined to determine arbitration status.

EFN1,0: Enhanced Function Bits 2,1

00 No operation—these bits ALWAYS read 00.

01 When this pattern is written the Parity and Interrupt latches are reset. This pattern should be followed by any other pattern to remove the reset (usually 00). This function replaces the RPI function performed when reading HA 7 in MODE N. Only the latches with ISR bits set to 1 will be reset.

10 Start DMA Initiator Receive. At the end of the write cycle the SDI function is performed. There is no need to follow with another pattern as required with the RPI value (01). This function replaces the SDI function performed when writing HA 7 in MODE N.

11 Read/Write ISR/IMR. When this pattern is written the NEXT read of HA 7 will access the ISR, or the NEXT write to HA 7 will access the IMR. This state ONLY lasts for the ONE following read OR write cycle. Further cycles will then access the EMR.

LOOP: Loopback Mode Bit 3

0 Normal operation

1 When set: SCSI drivers are disabled and the SCSI I/O's looped back inside the EASI, and both the TARGET and INITIATOR signals may be driven simultaneously. This enables the μP to check EASI operation without affecting the SCSI bus.

SPOL: SCSI Parity Polarity Bit 4

0 SCSI parity is ODD (as per SCSI specification).

1 SCSI parity is EVEN. This allows diagnostics to be performed.

3.0 Register Description (Continued)

MPOL: μP Parity Polarity Bit 5

0 μP parity is ODD.

1 μP parity is EVEN.

MPEN: μP Parity Enable Bit 6

0 μP parity checking and generation disabled.

1 Enable checking of μP data bus parity during μP and DMA writes. Generate parity during μP and DMA reads. Parity errors will cause an interrupt and set the ISR MPE bit if not masked.

APHS: Any Phase Mismatch Bit 7

0 Disable phase mismatch detection

1 Detect SCSI requests with a phase mismatch present. Is set when $\overline{REQ}$ goes active AND the SCSI phase lines do not match the contents of the TCR. Can be used in INITIATOR mode to interrupt on TARGET phase changes.

INTERRUPT STATUS REGISTER (ISR)

8 Bits HA 7 (EFN = 11) Read Only

This register is accessed during the first read cycle after the EFN bits in the EMR have both been set to 1. Once read, successive accesses of HA 7 go to the EMR. This register provides all interrupt status within one register. This is intended to make determination of interrupt sources easier in MODE E. In MODE N two registers must be read—the BSR and CSB. In MODE E only the ISR needs to be read. Additionally each interrupt (except SCSI $\overline{RST}$) has a corresponding status bit in the ISR.

When the ISR is read all unmasked, enabled, active interrupt sources set their corresponding ISR bits to a ONE. The interrupt status is sampled at the beginning of the ISR read cycle. When an RPI function is performed via the EMR, only each interrupt latch with an ISR bit set will be reset. This means interrupts occurring since the last ISR read will not be lost.

ISR bits may be individually masked via their corresponding bits in the IMR.

Bit 7							Bit 0
SPE	MPE	EDMA	DPHS	APHS	BSY	SEL	ARB

Interrupt Status Register

ARB: Arbitration Complete Bit 0

This bit is set when arbitration enabled by EMR ARB bit (0) has completed. Completion occurs in two ways: when the EASI loses arbitration and has asserted the LA bit in the ICR; or when the EASI has asserted the ID contained in the ODR onto the data bus, asserted $\overline{BSY}$, then waited for the 2.2 μs SCSI Arbitration Delay.

SEL: Selection/Reselection Bit 0

This bit is set when $\overline{BSY}$ is false, $\overline{SEL}$ is active, and any SER bit set to 1 has an active corresponding SCSI data bus bits. This situation occurs during Selection or Reselection phases.

BSY: Busy Loss Bit 2

This bit is the same as BSR bit 2. Set when the SCSI $\overline{BSY}$ signal becomes inactive while the MR2 BSY (monitor $\overline{BSY}$) bit is set.

APHS: Any Phase Mismatch Bit 3

Set when a $\overline{REQ}$ occurs while the SCSI phase lines do not match. Bit is set to enable detection of the contents of the TCR. EMR APHS (bit 7) must be 1 to allow this bit to be set.

DPHS: DMA Phase Mismatch Bit 4

Set when a SCSI DMA mode operation occurs with a phase mismatch. Similar to the APHS condition but restricted to DMA mode only. In MODE N this condition is not as easily determined.

EDMA: End of DMA Bit 5

Set when $\overline{REQ}$ and $\overline{ACK}$ are both false following the DMA cycle during which $\overline{EOP}$ was asserted. This represents the true end of DMA operation.

MPE: μP Parity Error Bit 6

Set when μP parity error is detected at the end of a μP or DMA write cycle. The MPEN bit in the EMR must be set for μP parity to be checked.

SPE: SCSI Parity Error Bit 7

Set when a SCSI parity error is detected when reading the CSD, during selection/reselection, or when the IDR is loaded during DMA operation. The MR2 PCHK bit must be set for SCSI parity to be checked, and the MR2 PINT bit must be set to enable the interrupt.

INTERRUPT MASK REGISTER (IMR)

8 Bits HA 7 (EFN = 11) Write Only

This register is accessed during the first write cycle after the EFN bits in the EMR have been set to 11. Once written, successive reads or writes at HA 7 access the EMR.

This register has the same bit definition as the ISR. If a bit in the IMR is set to 1 that interrupt will be masked. The interrupt will be captured internally (if enabled) but will not cause an active INT signal. A bit reset to 0 will enable that interrupt to occur (if enabled) and will enable the ISR bit to be set to 1.

Bit 7							Bit 0
SPE	MPE	EDMA	DPHS	APHS	BSY	SEL	ARB

Interrupt Mask Register

4.0 Device Operation

4.1 GENERAL

This section describes overall operation of the EASI. More detailed information of data transfers, interrupts and reset conditions are covered in later sections. The operation description covers μP accesses, SCSI bus monitoring, arbitration, selection, reselection, programmed-I/O, DMA interrupts. Programming and timing details are covered.

For information regarding interfacing to μPs and DMA controllers refer to Section 9.

In the descriptions following, program examples are given in pseudo-C. This processor-independent approach should be clearest. These are backed up by flow charts in Appendix A1.

For each section where appropriate the description is split into two, MODE N and MODE E.

4.0 Device Operation (Continued)

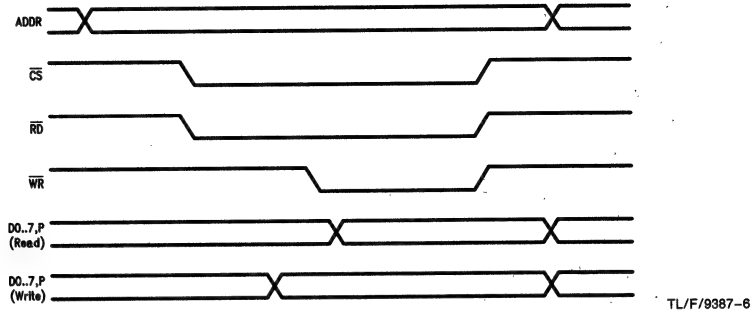


FIGURE 4.2. μP Cycles

4.2 μP ACCESSES

The μP accesses the EASI via the $\overline{CS}$, $\overline{RD}$, $\overline{WR}$ and address and data lines in order to read/write the registers. *Figure 4.1* shows typical timing. Note the use of non-multiplexed address and data lines.

4.3 SCSI BUS MONITORING/DRIVING

The SCSI bus may be monitored or driven at any time. Each bus signal is buffered and inverted by the EASI and can be read via the CSB, BSR and CSD registers. An active SCSI signal reads a 1 in the status registers.

Each SCSI signal may be asserted by setting a bit in the TCR or ICR. Setting the bit to 1 asserts the SCSI signal.

The following code demonstrates a byte transferred via programmed-I/O in INITIATOR mode.

```
{    /* Transfer one byte as Initiator */
    while (NOT (TCR: REQ)) ;
     /* wait till TARGET asserts REQ */
    data = input (CSD) ;
     /* parity is checked if enabled */
    output (ICR, Assert_ACK) ;
    while (TCR:REQ) ;
     /* wait till TARGET deasserts REQ */
    output (ICR, 0) ;
     /* deassert ACK, ready for next byte */
}
```

4.4 ARBITRATION

This sub-section describes the arbitration support provided by the EASI and how to program it.

4.4.1 MODE N Arbitration

Since the SCSI arbitration process requires signal sequencing too fast for μPs, hardware support is provided by the EASI. The arbitration process is enabled by bit 0 MR2 (ARB). Prior to setting this bit the ODR should be programmed with the device's SCSI ID—a single bit.

The EASI will monitor the bus for a BUS FREE phase. The $\overline{BSY}$ signal is continuously monitored. If continuously inactive for at least a SCSI Bus Settle Delay (400 ns) and $\overline{SEL}$ is inactive, a valid Bus Free Phase exists. After a period of SCSI Bus Free Delay (800 ns) the EASI asserts $\overline{BSY}$ and the ODR onto the SCSI data bus. The μP should poll the ICR to determine when arbitration has started. The AIP bit in the ICR is set when the Bus Free Phase is detected and the EASI is beginning the Bus Free Delay. Following the Bus Free Delay a 2.2 μs SCSI Arbitration Delay is required before examining the data bus to resolve the priorities of the ID bits. This delay must be implemented in firmware. The ICR Lost Arbitration (LA) bit must be examined to determine whether arbitration is lost. The LA bit is set if another device asserts $\overline{SEL}$ during arbitration. If the LA bit is 0 the data bus is read via the CSD register. The data is examined to resolve ID priorities. If this device is the highest ID, assert $\overline{SEL}$ by setting ICR bit 2 to a 1. After waiting Bus Clear + Bus Settle Delays (1200 ns), the Selection Phase begins. These 2 delays must be implemented in firmware.

4.4.2 MODE E Arbitration

The extended arbitration in MODE E is enabled by bit 0 of the EMR (ARB). This alternative offers two significant advantages over MODE N. First the 2.2 μs SCSI Arbitration Delay is implemented by the EASI. Second the arbitration process may be interrupt driven.

In MODE N the EASI must be polled to see when the ICR AIP bit is set. After the appropriate delays, the LA bit and data bus are examined. If arbitration is lost or this device is not the highest priority the MR2 ARB bit must be reset to 0, then set to 1 again and the whole process restarted. This means the EASI MUST be polled until arbitration is won—potentially many SECONDS, typically ms. This ties up the host μP.

In MODE E when the EMR ARB bit is set the EASI will: wait for BUS FREE phase; delay SCSI Bus Free Delay; assert $\overline{BSY}$ and the ODR onto $\overline{DB0}$–$\overline{DB7}$; delay SCSI Arbitration Delay; interrupt the μP.

The μP should read the ISR and if the ISR bit 0 (ARB complete) is set examine ICR bits 5 and 6 (LA and AIP) to determine whether arbitration is lost. If not lost the data bus is examined to resolve ID priorities. As for MODE N if arbitration has failed the EMR ARB bit should be reset and then set again after first resetting the ARB interrupt. Note that the EMR ARB bit allows the μP to carry on with other tasks while the EASI arbitrates. This means there is NO NEED to poll the EASI.

4.5 SELECTION/RESELECTION

The EASI can be used to select or reselect a device. The EASI will also respond to selection or reselection. Selecting or reselecting a device is the same in MODE N or MODE E. Response to selection or reselection can differ between the bus modes.

4.0 Device Operation (Continued)

4.5.1 Selecting/Reselecting

Selection requires programming the ODR with the desired and own device IDs; the data bus via ICR DBUS (bit 0); asserting $\overline{ATN}$ if required via ICR bit 1; asserting $\overline{SEL}$ via ICR bit 2; then resetting the MR2/EMR ARB bit.

The SER should have been cleared to zero before Selection/Reselection to ensure the EASI does not respond. If Reselection is desired the $\overline{I/O}$ line should also be asserted before $\overline{SEL}$ via TCR bit 0.

Resetting the ARB bit causes the EASI to remove $\overline{BSY}$ and the ODR from the data bus. Thus the ICR Assert data bus bit is required to assert the bits for desired and own device IDs.

$\overline{BSY}$ is then monitored to determine when the device has responded to (re)selection. If the device fails to respond an error handler should sequence the EASI off the bus. If the device responds the ICR DBUS and $\overline{SEL}$ bits should be reset to remove these signals. If this is a Reselection the ICR $\overline{BSY}$ bit (3) should be set before removing the other signals.

The bus is now ready to handle Information Transfer Phases.

4.5.2 MODE N (Re)Selection Response

The EASI responds to Selection or Reselection when the SER is non-zero. A (re)selected interrupt is generated when $\overline{BSY}$ is false for at least a Bus Settle Delay (400 ns); and $\overline{SEL}$ is true AND any non-zero bit in the SER has its corresponding SCSI data bus bit active. A Selection is disabled by zeroing the SER. If parity is supported it should be valid during (re)selection so it must be checked via the SPE bit (5) in the BSR. SCSI specification states that (re)selection is not valid if more than 2 data bits are active. This condition is checked by reading the CSD.

When the selection interrupt occurs it is determined by reading the BSR and CSB registers. There is no dedicated status bit for (re)selection in MODE N, it must be determined by the absence of other interrupts, and the active state of the $\overline{SEL}$ signal. Reselection occurs when $\overline{I/O}$ is also active. See Section 6.

4.5.3 MODE E (Re)Selection Response

The same conditions for valid (re)selection apply as for MODE N. A (re)selection interrupt is generated as per MODE N. The difference is that the interrupt sets the SEL bit (1) in the ISR. In MODE E, reading the ISR enables exact determination of interrupt sources. This interrupt can be masked via bit 1 of the IMR. The interrupt is reset by programming the RPI function to the EMR. See Section 6 for further details.

4.6 MONITORING BSY

While an INITIATOR is connected to a TARGET the TARGET must maintain an active $\overline{BSY}$ signal. During DMA operations the $\overline{BSY}$ signal is monitored by the EASI and will halt operations if it goes inactive. To enable $\overline{BSY}$ to be monitored at other times the MR2 BSY bit (2) should be set. An interrupt will be generated if $\overline{BSY}$ goes inactive while MR2 BSY is set.

In MODE N this interrupt sets bit 2 in the BSR. In MODE E this interrupt sets bit 2 in the BSR and bit 2 in the ISR. The interrupt may be masked via bit 2 of the IMR.

4.7 COMMAND/MESSAGE/STATUS TRANSFERS

Command, message and status bytes are transferred using programmed-I/O. The SCSI $\overline{REQ}/\overline{ACK}$ handshake is accomplished by monitoring and setting lines individually. Data is output via the ODR and read in via the CSD register.

The following code shows INITIATOR and TARGET programming for two of these cases. See Appendix A1 for flowcharts.

Initiator Command Send

```
{
  MR2 = monitor BSY
  TCR = Command Phase  /* 02h */
  while (bytes) to do) {
        while (REQ) inactive)
              idle; /* CSB bit 5 = 0 */
        if (BSR: phase_match==0)
              phase error;
        else  {
              ODR = data byte;
              ICR = Assert_ACK;
              while (REQ active)
              idle; /* CSB bit 5==1 */
              ICR = deassert_ACK
              /* byte transfer complete */
              byte count - -;
        }
  }
  goto data phase;
}
```

Target Message Receive

```
{
  /* assumed Assert_BSY already set in ICR */
  MR2 = TARG MODE OR PARITY CHECK
        OR PARITY INTERRUPT;
  TCR = Message_Out phase; /* 06h */
  delay (Bus Settle);
  TCR = Assert_REQ;
  while (ACK inactive)
        idle; /* BSR bit 0 */
  data = CSD; /* parity is latched */
  if (BSR: parity_error)
        error routine;
  else  {
        TCR = deassert_REQ;
        while (ACK active)
              idle;
  }
    /* message done, can change to next
       phase*/
}
```

4.8 NON-BLOCK DMA TRANSFERS

Data transfers may be effected by DMA. This method should be used for optimum performance. Two methods of DMA are available—block and non-block mode. This section describes non-block mode transfers. MODE N operation is covered first followed by MODE E.

The interface to the DMA controller uses the DRQ, $\overline{DACK}$, $\overline{EOP}$ lines in non-block mode. Each byte is requested (DRQ) and ack'd ($\overline{DACK}$). Representative timing for a DMA read is shown in *Figure 4.8.1*.

4.0 Device Operation (Continued)

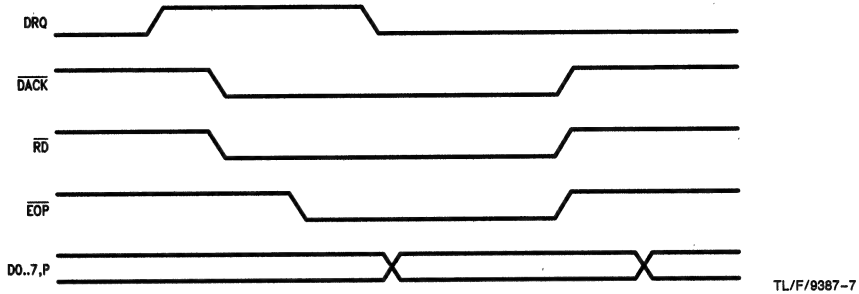


FIGURE 4.8.1. Non-Block DMA Timing

4.8.1 MODE N Non-Block DMA

DMA operation involves programming the EASI with the set-up parameters, initiating the DMA cycles and checking for correct operation when the completion interrupt is received. The DMA controller should be programmed with the data byte count and the memory start address. Methods of halting a DMA operation are covered in Section 4.11.

Setting up the EASI requires enabling or disabling the following: Data bus driving, DMA mode enable, $\overline{BSY}$ monitoring, $\overline{EOP}$ interrupt, parity checking, parity interrupt, TARGET Mode, bus phase.

Once set up, DMA should be initiated by writing to address 5, 6 or 7 as appropriate. The DMA controller should assert $\overline{EOP}$ during the transfer of the last byte, although this may be done by the μP if the DMA transfers (n−1) bytes and the μP transfers the last byte. See the application guide for more details (Section 9).

Upon completion the μP should check the following as required: End of DMA, Parity Error, Phase Match, Busy Error. In MODE N the end of DMA occurs as a response to $\overline{EOP}$. SCSI transfers may still be underway so $\overline{REQ}$ and $\overline{ACK}$ must still be checked to establish when the final byte is finished.

The code below shows programming of the EASI in each of the four DMA cases. One of these cases is shown in a flow diagram in Appendix A.

```
Initiator Send   /* DATA OUT PHASE */
{
  Program DMA Controller;
  TCR = 00h;     /* phase */
  ICR = 01h;     /* Assert_DBUS */
  MR2 = 0Eh;
  SDS = 00;      /* Start DMA Send */
  while (NOT interrupt)
        idle;
  while (CSD:REQ)
        idle;    /* wait for last SCSI byte
                    transfer so phase
                    is checked */
  if(BSR:Busy_error OR NOT(BSR:End_of_DMA))
        error routine;
  else {         /* DMA END */
        MR2 = 04h;  /* reset DMA bit */
        ICR = 0;
  }
}
```

```
Initiator Receive  /* DATA IN PHASE */
{
  Program DMA Controller;
  TCR = 01h;     /* phase */
  MR2 = 3Eh;
  SDI = 0;       /* Start DMA Init Rx */
  while (NOT interrupt)
        idle;
  /* no need to wait for last SCSI handshake
     done since DMA done implies it is
     checked */
  if (BSR:parity_error OR BSR:busy_error
        or NOT (BSR End of DMA))
              do error routines;
  else  {        /* End of DMA */
        while (CSD:REQ)
              idle;  /* wait for REQ inactive
                        to deassert ACK */
        MR2 = 04h;
  }
}

Target Receive  /* DATA OUT PHASE */
{
  Program DMA controller;
  TCR = 0;       /*phase*/
  ICR = 08h;
  MR2 = 7Ah;     /*check parity*/
  SDT = 0;       /*Start DMA Targ Rx*/
  while (not interrupt)
        idle;
  /*when End of DMA occurs the last byte
    has been read and checked*/
  if(BSR:parity_error OR NOT(BSR:End_of_DMA)
        error routine;
  else  {        /* End of DMA */
  while (BSR:ACK)
        idle;
  /*Not True End of DMA in MODE N, so wait
    until SCSI bus inactive before
    changing phase */
  MR2 = 40h;
  change phase as required;
  }
}
```

4.0 Device Operation (Continued)

```
Target Send  /* DATA IN PHASE */
{
  Program DMA Controller;
  TCR = 01h;   /* phase */
  ICR = 09h;
  MR2 = 4Ah;
  SDS = 0;     /* Start DMA Send */
  while (NOT interrupt)
        idle;
  if (NOT(BSR:END_of_DMA))
        error;
  else  {      /* DMA end */
        repeat {
              while (CSB:REQ OR BSR:ACK)
              loop count = 3;
              loop count - -;}
              /* decrement */
        until (loop count == 0);
        MR2 = 40h;
  Change phase as required;
  }
}
```

Some explanation of the final part of Target Send is required. In this type of DMA operation it is very difficult to exactly determine the True End of DMA in MODE N. Simply detecting $\overline{REQ}$ and $\overline{ACK}$ simultaneously inactive is not enough.

Reference to *Figure 4.8.2* will help to understand the following text.

As shown in *Figure 4.8.2*, $\overline{ACK}$ going active causes the DRQ for the next byte and also $\overline{REQ}$ to go inactive. $\overline{ACK}$ going inactive allows $\overline{REQ}$ to go active for the next byte. If the INITIATOR is slow removing $\overline{ACK}$ the μP may sample the SCSI bus after the $\overline{EOP}$ interrupt at point A. Here both $\overline{REQ}$ and $\overline{ACK}$ will be inactive, but there is one more byte to transfer on SCSI. Due to chip timing delays this condition will not last more than 200 ns. A safe way to determine the True End of DMA is to sample $\overline{REQ}$ and $\overline{ACK}$ and ONLY when both are inactive in three successive samples will the μP be at point B in the figure.

In MODE E, True End of DMA is correctly decoded and the End of DMA interrupt occurs as a result of this True condition, thus there is no need to sample $\overline{REQ}$ and $\overline{ACK}$. For this and other reasons operation in MODE E is strongly recommended.

4.8.2 MODE E Non-Block DMA

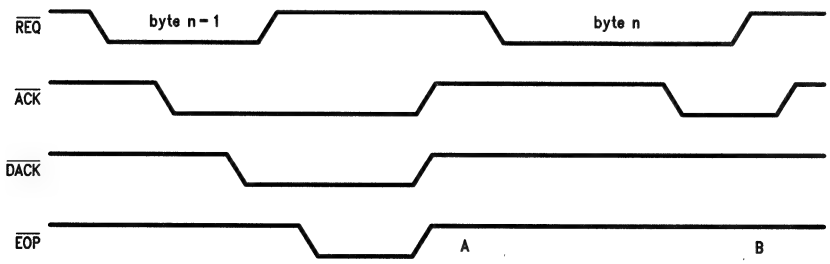
Operation for the Non-Block DMA in MODE E is essentially the same as MODE N. A primary difference is that in MODE E True End of DMA is decoded internally and this causes an interrupt. This feature removes the need to check for DMA End before moving to the next phase. Additionally, the use of the ISR allows easier determination of the end condition. For more information on interrupts see Section 6.

Examples of code are given below.

```
Initiator Receive  /* DATA IN PHASE */
{
  Program DMA controller;
  TCR = 01h;   /* phase */
  ICR = 40h;   /* MODE E */
  MR2 = BEh;
  EMR = 04h;   /* Start DMA INIT Rx */
  while (NOT interrupt)
        idle;
  if (ISR ! = 20h)
        error;
else          /* DMA end */
  MR2 = 04h;
}

Target Send    /* DATA IN PHASE */
{
  Program DMA controller;
  TCR = 01h;   /* phase */
  ICR = 49h;   /* MODE E */
  MR2 = CAh;
  SDS = 0;     /* Start DMA Send */
  while (NOT interrupt)
        idle;
  if (ISR! = 20h)
        error;
  else  {      /* DMA end */
  MR2 = 40h;
  Change phase as required;
  }
}
```

4.9 BLOCK MODE DMA TRANSFERS

In Block Mode the DMA interface uses the DRQ, $\overline{DACK}$, $\overline{EOP}$ and READY lines, DRQ is asserted once at the begin-



TL/F/9387–8

FIGURE 4.8.2. Target Send DMA

4.0 Device Operation (Continued)

ning of transfers and deasserted once $\overline{DACK}$ is received. $\overline{DACK}$ should be asserted continuously for the duration of all the transfer. $\overline{EOP}$ should be asserted during the last DMA byte signal when the next DMA byte transfers. The EASI asserts the READY signal when the next DMA byte should be transferred. As for non-block mode the End of DMA interrupt is just $\overline{EOP}$ in MODE N, but is a True End of DMA in MODE E.

The block mode is intended for systems where the overhead of handing the system busses to and from the μP and DMA controller is too great. The block mode handshake is not necessarily faster than non-block (it may be) but the overall transfer rate is improved once the bus exchange overhead is removed. Of course the μP is prevented from executing for the whole DMA operation.

If a phase mismatch occurs the READY signal is left in the inactive state. The DMA controller must hand back the bus to the μP and the inactive READY signal may need to be gated off. For more detail see Section 9.

When performing DMA as an INITIATOR the $\overline{EOP}$ signal does not deassert $\overline{ACK}$ on the SCSI bus in MODE N. Firmware must determine when $\overline{REQ}$ is inactive after the last SCSI transfer then reset the MR2 DMA bit to deassert $\overline{ACK}$. In MODE E the EASI correctly handles the deassertion of $\overline{ACK}$ after True End of DMA.

Programming the EASI in block mode is the same as non-block mode except bit 7 in MR2 should also be set.

4.10 PSEUDO DMA

The system design can utilize EASI DMA logic for non data transfers. This removes the need to poll $\overline{REQ}$/$\overline{ACK}$ and program the assertion/deassertion of the handshake signal. The μP can emulate a DMA controller by asserting $\overline{DACK}$ and $\overline{EOP}$ signals. DRQ may be sampled by reading the BSR. In most cases the chip decode logic can be adapted to this use for little or no cost. See Section 9 for further details.

4.11 HALTING A DMA OPERATION

There are three ways to halt a DMA operation apart from a chip or SCSI reset. These methods are: $\overline{EOP}$, phase mismatch and resetting the DMA MODE bit in MR2.

4.11.1 End of Process

$\overline{EOP}$ is asserted for a minimum period during the last DMA cycle. The $\overline{EOP}$ signal generates the End of DMA interrupt in MODE N, and enables the True End of DMA to occur later in MODE E. $\overline{EOP}$ does not cause the MR2 DMA mode bit to be reset.

4.11.2 DMA Phase Mismatch

If a $\overline{REQ}$ goes active while there is a phase mismatch the DMA will be halted and an interrupt generated. The EASI will stop driving the SCSI bus when the mismatch occurs. A phase mismatch is when the TCR phase bits do not match the SCSI bus values.

4.11.3 DMA Mode Bit

If $\overline{EOP}$ is not used, the best method is to reset the MR2 DMA Mode bit. This bit may be reset at any time, and should be reset after an End of DMA interrupt or a phase mismatch. Resetting the bit disables all DMA logic and thus should only be reset at the True End of DMA condition. Additionally, all DMA logic is reset, so this bit must be reset then set again to carry out the next DMA phase.

5.0 Interrupts

5.1 OVERVIEW

The EASI is intended to be used in an interrupt driven environment. MODE E has greatly enhanced the use of interrupts to relieve firmware overhead. This section describes the conditions for and use of each interrupt. Each description explains MODE N then MODE E operation.

Before individually describing each interrupt, an explanation of the use of interrupts is required.

5.2 USING INTERRUPTS

MODE N: In this mode interrupts are controlled by bits in MR2 if control is provided. Not all interrupts can be disabled under software control. When an interrupt occurs, both the BSR and CSD register must be read and analyzed to determine the source of interrupt. Since status is NOT provided for each interrupt, great care should be exercised when determining the interrupt source.

MODE E: In this mode every interrupt can be individually masked and enabled or disabled. In addition, when an interrupt occurs a single register (ISR) can be read to determine the source(s) of interrupt. An associated register (IMR) allows a mask bit to be set for each interrupt. Finally, the design of the logic prevents loss of interrupts occurring after the INT signal goes Active and before the Reset Parity/Interrupt function. In MODE N loss of interrupts can occur.

5.3 SCSI PARITY ERROR

MODE N: If SCSI parity checking is enabled via MR2 bit 5, an interrupt can occur as a result of a read from CSD, a selection/(re)selection, or a DMA receive operation. The parity error bit (bit 5) in the BSR will be set if checking is enabled. An interrupt will occur if Enable Parity Interrupt (bit 4) of MR2 is set. The interrupt is reset by reading HA 7. Following an interrupt the BSR and CSB should contain the values shown below.

Bit 7 **Bit 0**

X	X	1	1	X	X	X	X
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7 **Bit 0**

0	1	X	X	X	X	0	X
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSB

5.4 μP PARITY ERROR

MODE N: μP parity is not available under this mode.

MODE E: If bit 6 of the EMR is 1, μP parity will be checked on a write to the EASI via the μP or DMA controller. The parity polarity is determined by bit 5 of the EMR. If bit 6 of the IMR is zero, a μP parity error will set bit 6 of the ISR and cause an interrupt. The interrupt may be reset by writing 01 to bits 2 and 1 of the EMR followed by any other pattern.

5.5 END OF DMA

MODE N: If $\overline{EOP}$ is asserted during a DMA transfer, bit 7 of the BSR will be set and an interrupt generated if bit 3 of MR2 is 1. $\overline{EOP}$ is recognized when $\overline{EOP}$, $\overline{DACK}$ and either $\overline{IOR}$ or $\overline{IOW}$ are all simultaneously active for a minimum period. The interrupt may be reset by reading HA 7. Following an interrupt the BSR and CSB should contain the values shown below.

5.0 Interrupts (Continued)

Bit 7							Bit 0
1	X	X	1	X	X	0	X
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	1	X	X	X	X	0	X
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSB

5.6 DMA PHASE MISMATCH

MODE N: When the SCSI $\overline{REQ}$ goes active during a DMA operation the contents of the TCR are compared with the SCSI phase lines C/$\overline{D}$, $\overline{MSG}$ and I/$\overline{O}$. If the two do not match an interrupt is generated. This interrupt will occur as long as the MR2 DMA bit is set (bit 1), i.e., it cannot be masked in MODE N. The mismatch removes the EASI from driving the SCSI data bus. The interrupt may reset by reading HA 7. Following an interrupt the BSR and CSB should contain the values shown below.

Bit 7							Bit 0
X	0	X	1	0	X	X	X
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	X	X	X	X	X	0	X
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSB

MODE E: Bit 4 of the ISR will be set by a DMA phase mismatch—the same conditions as MODE N. The interrupt and setting if ISR bit 4 may be masked by setting bit 4 of the IMR. The interrupt may be reset by writing 01 to bits 2 and 1 of the EMR followed by any other pattern.

5.7 ANY PHASE MISMATCH

MODE N: This feature is not available under this mode.

MODE E: This condition is similar to DMA Phase Mismatch except that it applies to all operations—not just DMA. If the TCR contents do not match the SCSI phase lines when $\overline{REQ}$ goes active an interrupt is generated and bit 3 set in the ISR. The ISR bit and the interrupt may be masked by setting bit 3 in the IMR. The interrupt may be reset by writing 01 to bits 2 and 1 of the EMR followed by any other pattern.

5.8 BUSY LOSS

MODE N: If bit 2 in MR2 is set the SCSI $\overline{BSY}$ signal is monitored and an interrupt is generated if $\overline{BSY}$ is continuously inactive for at least a BUS SETTLE DELAY (400 ns). This interrupt may be reset by reading HA 7. Following an interrupt the BSR and CSB should contain the values shown below, where usually CSB = 00.

Bit 7							Bit 0
X	X	X	1	X	1	X	X
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	0	X	X	X	X	X	X
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSB

MODE E: Bit 2 in MR2 performs the same function as in MODE N. When an interrupt is generated bit 2 in the ISR will be set. The ISR bit and the interrupt may be masked by setting bit 2 in the IMR. The interrupt may be reset by writing 01 to bits 2 and 1 of the EMR followed by any other pattern.

5.9 (RE)SELECTION

MODE N: An interrupt will be generated when: $\overline{SEL}$ is active, $\overline{BSY}$ is inactive, and the device ID is true. The device ID is determined by the value in the SER. If ANY non-zero bit in the SER has its corresponding SCSI data bit active during selection, the device ID is true. If I/$\overline{O}$ is active this is a reselection. The interrupt is disabled by writing all zeroes to the SER, and reset by reading HA 7.

If SCSI parity checking is enabled it will be checked and should be valid. Following an interrupt the BSR and CSB should contain the values shown below.

Bit 7							Bit 0
0	0	0	1	X	0	X	0
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	0	0	0	0	0	1	X
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSB

MODE E: The functioning of the (re)selection logic is the same as for MODE N. Additionally, bit 1 in the ISR will be set upon a (re)selection interrupt. The interrupt and setting of bit 1 in the ISR may be masked by setting bit 1 in the IMR. The interrupt may be reset by writing all zeroes to the SER then writing 01 to bits 2 and 1 of the EMR followed by any other pattern.

5.10 ARBITRATION COMPLETE

MODE N: No interrupt is generated in MODE N.

MODE E: When bit 0 of the EMR is set to 1 the EASI monitors $\overline{BSY}$ and $\overline{SEL}$ to determine a BUS FREE Phase. The EASI then carries out all steps required for bus arbitration. When either arbitration is lost or the arbitration process has completed, bit 0 in the ISR is set and an interrupt generated. The interrupt and bit 0 in the ISR can be masked by setting bit 0 in the IMR. Note that arbitration is not affected by the IMR, just the interrupt.

6.0 Reset Conditions

6.1 GENERAL

There are three ways to reset the EASI; μP chip $\overline{RESET}$, SCSI bus reset applied externally, SCSI bus reset issued by the EASI.

6.2 CHIP RESET

When the $\overline{RESET}$ signal is asserted for the required duration the EASI clears ALL internal registers and therefore resets all logic. This action does not create an interrupt or generate a SCSI reset. Since all registers contain zeroes,

6.0 Reset Conditions (Continued)

the EASI is in MODE N under any of the three reset conditions.

6.3 EXTERNAL SCSI RESET

When an SCSI $\overline{RST}$ is applied externally the EASI resets all registers and logic and issues an interrupt. The only register bits not affected are the Assert RST bit (bit 7) in the ICR and the TARGET Mode bit (bit 6) in MR2. Note that the ISR will contain all zeroes.

6.4 SCSI RESET ISSUED

When the μP sets the Assert RST bit in the ICR, the $\overline{RST}$ signal goes active. Since the EASI monitors $\overline{RST}$ also, the same reset actions as in 6.2 apply. The SCSI $\overline{RST}$ signal will remain active as long as bit 7 in the ICR is set—i.e., until programmed 0 or a chip $\overline{RESET}$ occurs.

7.0 Loopback Testing

7.1 GENERAL

The DP8490 EASI features loopback testing, enabled by bit 3 of the EMR. When the LOOP bit is set in the EMR all SCSI drivers are disabled at the pads and the signals looped back internally. Additionally, both TARGET and INITIATOR signals may be simultaneously asserted—this is not possible in normal operation. All this enables testing of EASI operation without affecting the SCSI bus.

7.2 SCSI SIGNAL DRIVING/MONITORING

Since each SCSI signal is looped back, testing is accomplished by asserting a signal via the ICR or TCR, then reading back via CSB and BSR to check its state. The code below provides examples. The first example tests SCSI control signals and the second the SCSI data bus. Note that loopback mode must be enabled prior to asserting any signals if they are not to drive the SCSI bus.

```
SCSI_signal_test
{
  ICR = 40h;   /* MODE E */
  EMR = 08h;   /* LOOPback */
  ICR = 5Eh;   /* drive INIT controls */
  TCR = 0Fh;   /* drive TARG controls */
  if((CSB==7Eh) AND (BSR==0Fh))
        ok;
  else error;
}

SCSI_data_test
{
  ICR = 40h;   /* MODE E */
  EMR = 08h;   /* LOOPback */
  ICR = 41h;   /* assert data bus */
  MR2 = 30h;   /* parity check & interrupt */
  ODR = 0AAh;
  if(CSD==0AAh)  ok;
  else  error;
  ODR = 55h;
  if(CSD==55h)  ok;
  else  error;
  if(interrupt)error;  /* no parity errors */
  else ok;
}
```

7.3 (RE)SELECTION AND ARBITRATION

Both these features may be tested in loopback. Note that when checking BSY via the CSB register the "debounced" version of BSY is presented and will be active for 400 ns–800 ns after BSY goes inactive. Logic within the EASI continuously monitors BSY when it becomes inactive to detect a valid Bus Free Phase. One of the outputs of this logic is a clean version of BSY which is accessed through the CSB.

7.4 DATA TRANSFERS

Both programmed-I/O and DMA transfers may be performed. When doing DMA transfers the MR2 TARGET MODE bit (6) must be programmed according to the type of DMA—i.e., set to 1 for TARGET Send or Receive, reset to 0 for INITIATOR Send or Receive. Additionally, the actions of the other SCSI device must be programmed. For example, when testing INITIATOR operations the BSY signal must be set via ICR to simulate a TARGET connected, and the REQ signal must be programmed active and inactive to perform the handshake. The code below shows a single byte transfer as INITIATOR Send.

```
DMA_test
{
  program DMA controller;
  ICR = 40h;    /* MODE E */
  EMR = 08h;    /* LOOPback */
  ICR = 49h;    /* BSY & data bus on */
  MR2 = 0Ah;    /* DMA & EOP interrupt */
  TCR = 08h;    /* assert REQ */
  SDS = 0;      /* Start DMA Send */
  /* DMA cycle with EOP is done here */
  TCR = 0;      /* deassert REQ */
  if(IDR==data byte)  ok;
  else  error;
  if(BSR==90h)  ok;
  else  error;
  /* also should be an EOP interrupt */
}
```

8.0 Extra Features/Compatibility

8.1 OPERATING MODES

This section is intended to clearly indentify the differences between the DP8490 and DP5380. The description covers registers, signals, timing, "bugs" and enhancements. For a more detailed description of register programming and bit functions refer to Sections 3–7.

Before discussing differences a review of DP8490 operation is required. The EASI can be operated in two modes—Normal Mode (MODE N) and Enhanced Mode (MODE E). The EASI is in one or the other mode at any time. MODE E is selected when bit 6 in the INITIATOR COMMAND REGISTER (ICR) is set to 1. A SCSI or external chip reset clears all registers (except MODE REGISTER 2 bit 6) so MODE N is selected as the default. MODE N may also be selected at any time by clearing bit 6 of the ICR to zero. If MODE E

8.0 Extra Features/Compatibility (Continued)

functions have been invoked, selecting MODE N does not in general affect their operation. This means MODE E is used to enable selection of enhanced features which can be used in either mode. Normally an application will choose to remain in MODE E always since all functions are accessible in this mode. MODE N is only required for downward compatibility with a standard 5380 device. The DP5380 operates ONLY in MODE N.

The MODE selection via bit 6 of the ICR enables existing 5380 firmware to run unaltered with the EASI. On the 5380 this is a TEST MODE bit which disables ALL output drivers—NOTHING ELSE! No logic is "tested" and no changes are made to internal logic. Since the μP data bus drivers are also disabled the internal state of the 5380 is inaccessible. This means the TEST MODE bit has very limited application for the end user. By comparison the LOOPBACK bit in the EASI enables thorough testing of device operation.

In summary, current operating firmware should not be using bit 6 of the ICR so the DP8490 uses this bit to enable enhanced operation. *As long as this bit is not set in the EASI the DP8490 appears the same as the NCR, AMD or DP5380. Programming, device operation, and timing sequences are all the same as a 5380.*

8.2 INTERNAL REGISTERS

Figure 8.1 shows the register map of the EASI. Note that in MODE N hex address 7 (HA 7) causes a Parity/Interrupt Reset when read and a Start DMA Initiator Receive when written. In MODE E the read/write EXTRA MODE REGISTER (EMR) is mapped into HA 7. Since the two original functions at HA 7 would conflict, their functions are reproduced by writing to bits 1 and 2 of the EMR. This removes any need to switch between modes. The INTERRUPT MASK REGISTER (IMR) and the INTERRUPT STATUS REGISTER (ISR) are accessed indirectly via the EMR. Setting both bits 1 and 2 of the EMR to 1 enables the next read of address 7 to access the ISR or the next write to access the IMR. Once the access is made subsequent uses of address 7 use the EMR.

Hex Addr	Read Register	Write Register
00	Current SCSI Data	Output Data Register
01	Initiator Command	Initiator Command
02	Mode Register 2	Mode Register 2
03	Target Command	Target Command
04	Current SCSI Bus Status	Select Enable Register
05	Bus and Status	Start DMA Send
06	Input Data Register	Start DMA Targ Rx
07	Reset Parity/Interrupt (MODE N)	Start DMA Init Rx (MODE N)
07	Extra Mode Register (MODE E)	Extra Mode Register (MODE E)
07	Interrupt Status Reg (MODE E and bits 1 and 2 of EMR = 1)	Interrupt Mask Reg (MODE E and bits 1 and 2 of EMR = 1)

FIGURE 8.1. EASI Register Map

8.3 ENHANCEMENT MODE REGISTERS

MODE E provides three new registers to provide control for the extra features of the EASI. The EXTRA MODE REGISTER (EMR) is a read/write register which enables or disables the extra functions. Setting an EMR bit to 1 enables the corresponding function while programming a 0 disables it. Note on power-up or reset the EMR is zeroed so all extra functions are disabled. The INTERRUPT MASK REGISTER (IMR) provides the ability to individually mask out interrupts. The INTERRUPT STATUS REGISTER (ISR) shows the status of the interrupt system at any time. When an interrupt occurs the ISR will contain 1's for interrupts that are active, enabled and not masked. *Figure 8.2* shows the format of the three registers.

8.4 EMR FEATURES

8.4.1 Arbitration

In MODE N, arbitration is enabled via bit 0 of MR2. However this requires polling the device for potentially many milliseconds. The μP polls the ICR for the ARBITRATION IN PROGRESS bit (6) which is set once the ASI/EASI detects a bus free phase and enters arbitration. The μP must then wait for the 2.2 μs ARBITRATION DELAY before checking IDs on the bus. No interrupt is given for any of these events.

In MODE E, EASI arbitration is enabled by bit 0 of either MR2 or EMR. The EMR bit enables extended arbitration. The EASI will wait for bus free, arbitrate, wait the 2.2 μs ARBITRATION DELAY and the INTERRUPT the μP if not masked in the IMR. An interrupt occurs if arbitration is complete or has been lost. This feature removes the need to poll the EASI.

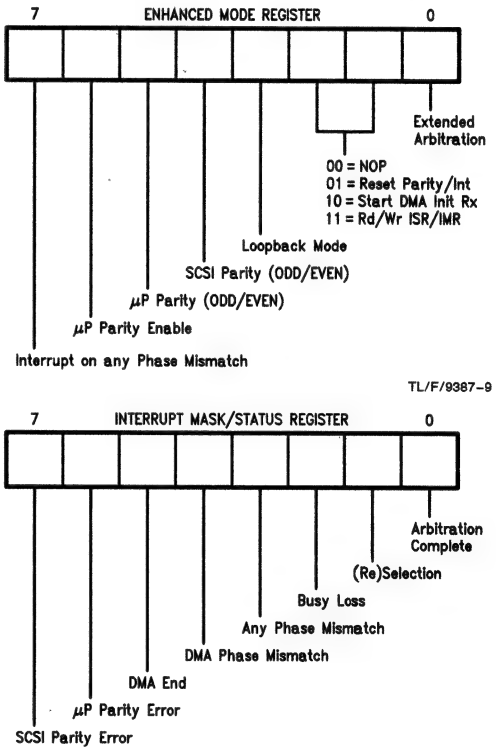


FIGURE 8.2. MODE E Registers

8.0 Extra Features/Compatibility (Continued)

8.4.2 Loopback Mode

When bit 3 is set in the EMR the EASI disables all SCSI drivers and loops back the signals internally. SCSI Data Out is linked to SCSI Data In. $\overline{BSY}$, $\overline{SEL}$, $\overline{ATN}$, $\overline{RST}$, $\overline{I/O}$, $\overline{C/D}$, $\overline{MSG}$, $\overline{REQ}$ and $\overline{ACK}$ outputs are fed back to their own inputs. This enables testing of EASI operations, including DMA.

8.4.3 SCSI Parity

SCSI parity may be enabled and also cause an interrupt (or be masked). Bit 4 of the EMR allows the polarity of SCSI parity to be set to either ODD or EVEN, with a default of ODD. This feature allows checking of SCSI devices and cable. It also allows an INITIATOR to automatically detect whether a device supports parity. (Send a parity error and check it is reported).

8.4.4 μP Parity

The EASI includes parity for the μP data bus. This enables controllers to validate data while it passes through their data buffers. In common with SCSI parity the μP parity may be: enabled/disabled, interrupt or be masked, be EVEN or ODD polarity.

8.4.5 Phase Mismatch

In MODE N the EASI will only give a Phase Mismatch interrupt during DMA (normally data) transfers. In MODE E an interrupt can also be programmed for a phase mismatch during any phase. The mismatch is detected if $\overline{REQ}$ is active and the phase lines do not match the phase expected as specified in the TCR bits 0 to 2. This feature allows completely interrupt-driven operation of the EASI.

8.5 INTERRUPTS

The EASI has an improved interrupt structure to ease programming. In MODE N the structure is the same as a standard 5380. In this mode interrupts can occur which may only be determined by the lack of other interrupts (e.g., Selection and Phase Mismatch). Interrupts can also be missed if they occur while servicing others. To determine the cause of an interrupt requires reading the BSR and CSB registers then interpreting the results, knowing what the 5380 was currently doing.

In MODE E interrupts can be individually masked via the IMR and active ones determined by reading the ISR. Any unmasked interrupts sets an ISR bit to 1. After the interrupt service routine the μP should perform a Reset Parity/Interrupt function via the EMR—this will ONLY RESET interrupts which had a 1 in the ISR when it was last read. This feature means interrupts will not be lost.

8.6 TIMING

The NCR5380 timing has some aspects which have been improved for MODE E of the EASI. In MODE N timing details are the same as the 5380. Of course the DP5380 ASI remains the same as the NCR5380. The timing improvements are listed below.

1. **True End of DMA:** In MODE N the end of DMA is when the last byte is transferred by the DMA controller, not when the final SCSI transfer is done. In MODE E bit 7 of the TCR shows a true end of DMA status—i.e., the last byte transferred by the later of the two events. This bit is compatible with the NCR CMOS 53C80.
2. **SCSI Handshake after EOP:** in MODE N, INITIATOR receive when a $\overline{REQ}$ is received after $\overline{EOP}$ has been given an $\overline{ACK}$ will be generated although no valid data exists since no DRQ was issued. The EASI will NOT generate this invalid $\overline{ACK}$ while in MODE E.

μP and DMA accesses have relaxed timing on both the EASI and ASI. Data setup/hold and read access times are reduced. Faster handshaking on the SCSI bus, along with faster response to the DMA signals means that higher transfer rates are possible with both devices—typically over 3 MBytes/s.

9.0 Application Guide

This section is intended to show the interface between the μP, EASI and DMA controller (DMAC). *Figure 9.1* shows a general interface when the EASI and DMAC are I/O-mapped devices. This configuration will implement a 2 to 2.5 MBytes/s SCSI port using 2 cycle compressed timing from the 5 MHz DMAC.

Using a faster DMAC and memory may allow the EASI to operate at a higher rate—but of course any system will be limited by the available DMA rate from the SCSI device currently connected to. The interface shown has several features that are examined more closely in the following text.

All the interface signal requirements are satisfied by a PAL device. The memory interface is not shown, only the relevant DMAC and μP lines are included.

The EASI data and address lines connect directly to the μP/DMAC busses. The DRQ output from the EASI goes direct to the DMAC. The $\overline{EOP}$ output from the DMAC goes to the EASI input, via the PAL, but can also be asserted via the PAL since the DMAC output is open-drain.

The PAL is programmed so that the μP can access the EASI in three ways. The three access types are: Register R/W, DMA R/W, DMA with $\overline{EOP}$. Examination of the PAL equations below shows how the μP may perform any of the three basic access types simply by accessing the EASI at different I/O address slots. This enables the μP to simulate a DMAC (pseudo-DMA). DMA mode may then be used for all information transfer phases.

In DMA mode the EASI generates all SCSI handshakes. At all other times the μP is responsible for $\overline{REQ}/\overline{ACK}$ handshakes. Using pseudo-DMA may reduce μP overhead.

When doing DMA transfers via BLOCK MODE and an error occurs, the EASI may not deassert the READY signal. For some DMA controllers this may lock the bus, so the PAL asserts READY and $\overline{EOP}$ to the DMA if an interrupt occurs while READY is false. This completes the current DMA cycle and prevents further DMA for the rest of the block thus allowing the bus to be handed back to the μP for servicing.

The PAL generates $\overline{IOR}$ and $\overline{IOW}$ strobes while the μP is bus master, but the DMAC provides the strobes while it is bus master so the PAL outputs are TRI-STATE.

The PAL details are shown in *Figure 9.2* with the signal definitions and equations following.

9.0 Application Guide (Continued)

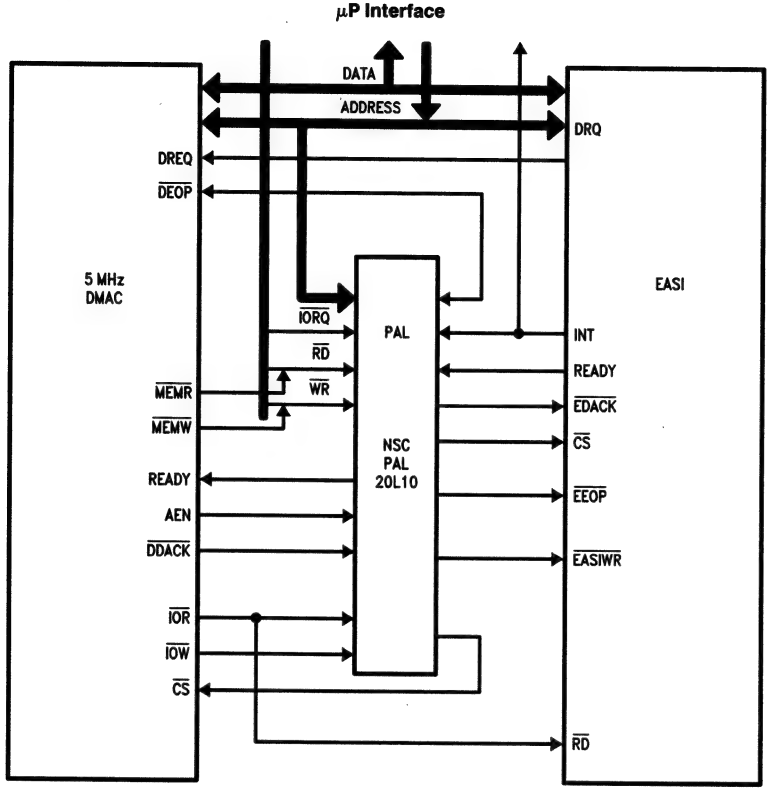


FIGURE 9.1. μP/EASI/DMA Interface

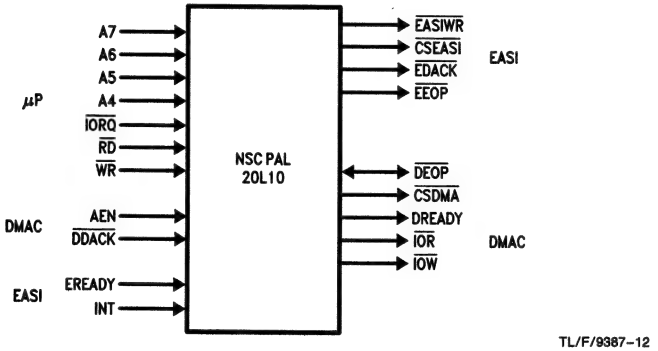


FIGURE 9.2. Interface PAL

9.0 Application Guide (Continued)

```
/CSEASI = /IORQ*/A7*/A6*/A5*/A4*/AEN          ; EASI reg R/W chip select
/EDACK =  /IORQ*/A7*/A6*/A5* A4*/RD           ; µP pseudo-DMA cycle
          /IORQ*/A7*/A6*/A5* A4*/WR
          +/IORQ*/A7*/A6* A5*/A4*/RD          ; µP pseudo-DMA with EOP
          +/IORQ*/A7*/A6* A5* A4*/WR
          +/DDACK                             ; DMAC DMA cycle
/EEOP = /IORQ*/A7*/A6* A5*/A4*/RD*/AEN        ; µP pseudo-DMA with EOP
        +/IORQ*/A7*/A6* A5*/A4*/WR*/AEN
        +/DEOP*EREADY                         ; Prevents EASI from seeing
                                              ; EOP until READY goes high.
IF(/DDACK*/EREADY*INT)/DEOP = /DDACK*/EREADY*INT; on error this will terminate the DMA
transfer.
/CSDMA = /IORQ*/A7*/A6*A5*A4                  ; DMAC register R/W
/DREADY = /EREADY*/INT                        ; EASI not READY and not INT
        +/EREADY*/DDACK                       ; EASI not READY and DMA cycle active
IF(/AEN) /IOR   = /IORQ*/RD                   ; µP I/O Read cycle
IF(/AEN) /IOW   = /IORQ*/WR                   ; µP I/O Write cycle
/EASIWR = /IORQ* WR*/AEN+/IOW*EREADY*AEN      ; Prevents SCSI data being
                                              ; changed before EASI is
                                              ; READY for next byte.
```

FIGURE 9.3. PAL Equations

The µP and DMA signals are defined below

A7–A4	Address bus
$\overline{\text{IORQ}}$	Memory I/O cycle select
$\overline{\text{RD}}$	Read Strobe
$\overline{\text{WR}}$	Write Strobe
AEN	High DMA address enable asserted by DMAC
$\overline{\text{DDACK}}$	DMAC DMA Acknowledge
$\overline{\text{CSDMA}}$	DMA Chip Select
DREADY	Ready signal to DMAC—inserts wait-states when low
$\overline{\text{IOR}}$, $\overline{\text{IOW}}$	I/O data strobes to/from DMAC
$\overline{\text{EASIWR}}$	EASI write strobe.

10.0 Absolute Maximum Ratings*

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7.0V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} + 0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} + 0.5V
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D)	500 mW
Lead Temperature (T_L) (Soldering, 10 seconds)	260°C
Electro-Static Discharge Rating	2 kV

*Absolute maximum ratings are those values beyond which damage to the device may occur.

11.0 DC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = 5.0V ±5% unless otherwise specified

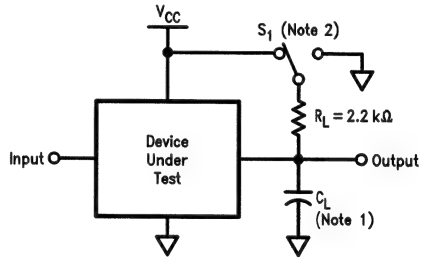
Symbol	Parameter	Conditions	Typ	Limit	Units
V_{IH}	Minimum High Level Input Voltage			2.0	V
V_{IL}	Maximum Low Level Input Voltage			0.8	V
V_{OH1} V_{OH2}	Minimum High Level Output Voltage	$\|I_{OUT}\|$ = 20 μA $\|I_{OUT}\|$ = 4.0 mA		V_{CC} − 0.1 2.4	V V
V_{OL1} V_{OL2} V_{OL3}	Maximum Low Level Output Voltage	SCSI Bus Pins: $\|I_{OL}\|$ = 48 mA Other Pins: $\|I_{OL}\|$ = 20 μA $\|I_{OL}\|$ = 8.0 mA		0.5 0.1 0.4	V V V
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND		±1	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current	V_{OUT} = V_{CC} or GND		±10	μA
I_{CC}	Supply Current	V_{IN} = V_{CC} or GND SCSI Inputs = 3V	2.5	4	mA

Capacitance T_A = 25°C, f = 1 MHz

Symbol	Parameter (Note 3)	Typ	Units
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	7	pF

AC Test Conditions

Input Pulse Levels	GND to 3.0V
Input Rise and Fall Time	6 ns
Input/Output Reference Levels	1.3V
TRI-STATE Reference Levels (Note 2)	active low + 0.5V
	active high − 0.5V



TL/F/9387–13

Note 1: C_L = 50 pF including jig and scope capacitance.

Note 2: S1 = Open for push-pull outputs.
S1 = V_{CC} for active low to TRI-STATE.
S1 = GND for active high to TRI-STATE.

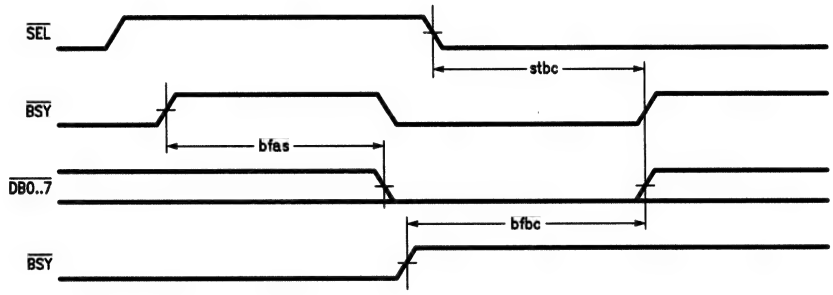
Note 3: This parameter is not 100% tested.

12.0 AC Electrical Characteristics

All parameters are preliminary and subject to change without notice

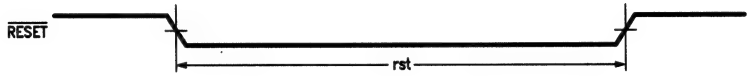
Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
bfas	BSY false to arbitrate start	1200		2200	ns
bfbc	BSY false to bus clear			800	ns
stbc	SEL true to bus clear			500	ns
rst	RESET pulse width	100			ns

12.1 ARBITRATION



TL/F/9387–14

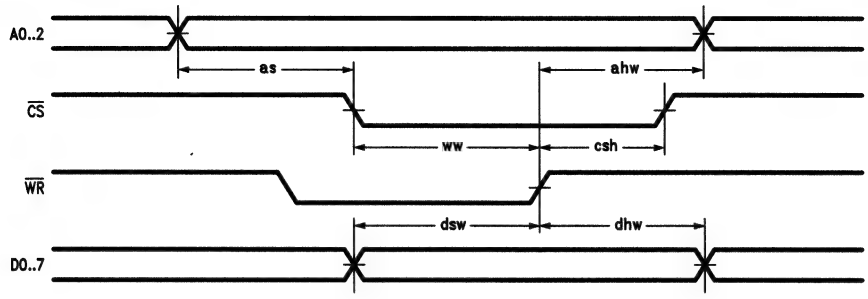
12.2 μP RESET



TL/F/9387–15

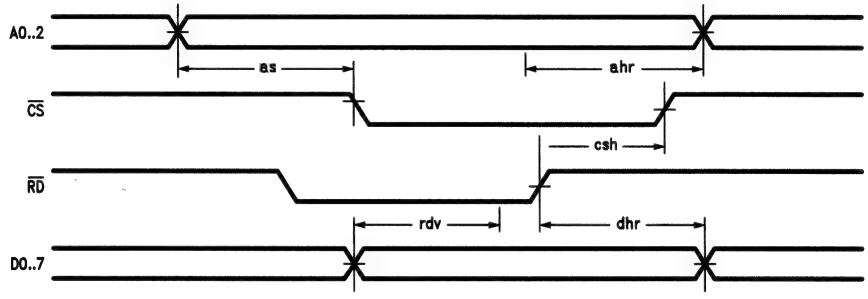
12.0 Electrical Characteristics

12.3 μP WRITE



TL/F/9387–16

12.4 μP READ



TL/F/9387–17

Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
ahr	Address Hold from and of Read Enable (Note 1)	5			ns
ahw	Address Hold from End of Write Enable (Note 2)	5			ns
as	Address Setup to Read or Write Enable (Notes 1,2)	5			ns
csh	CS Hold from End of RD or WR	0			ns
dhr	Data Hold from End of Read Enable (Notes 1,3)	20		60	ns
dhw	μP Data Hold Time from End of WR	10			ns
dsw	Data Setup to End of Write Enable (no μP Parity) (Note 2) (with μP Parity)	35 35			ns ns
rdv	Data Valid from Read Enable (Note 1)			50	ns
ww	Write Enable Width (Note 2)	40			ns

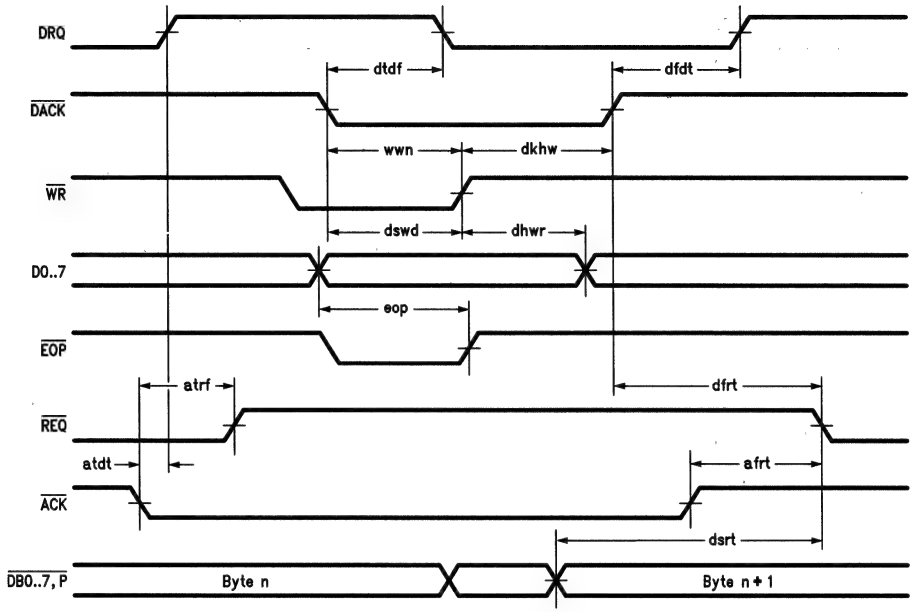
Note 1: Read enable (μP) is CS and RD active.

Note 2: Write enable (μP) is CS and WR active.

Note 3: This includes the RC delay inherent in the test's method. These signals typically turn off after 25 ns, enabling other devices to drive these lines with no contention.

12.0 AC Electrical Characteristics (Continued)

12.5 DMA WRITE (NON-BLOCK MODE) TARGET SEND



TL/F/9387–18

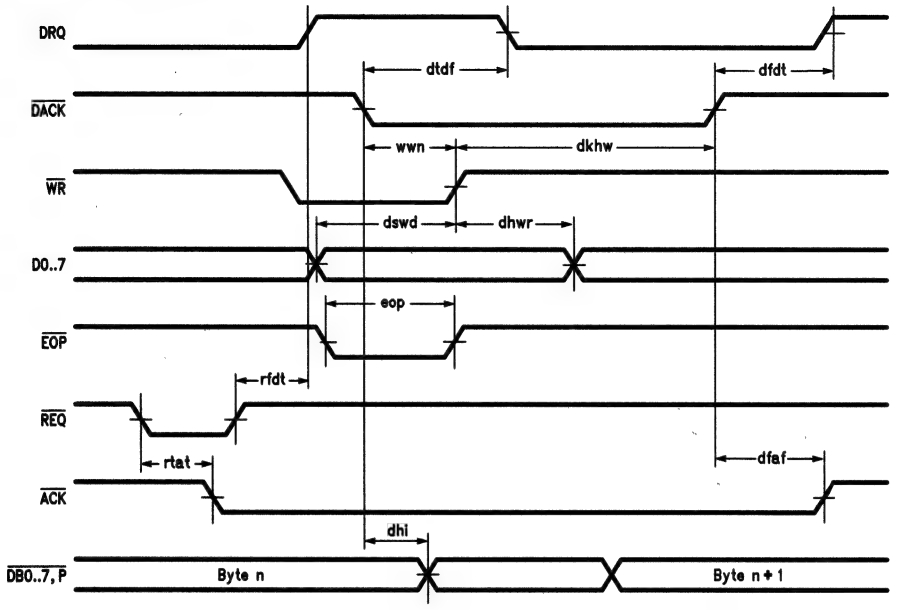
Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
afrt	ACK False to REQ True (DACK or WR False)			75	ns
atdt	ACK True to DRQ True			55	ns
atrf	ACK True to REQ False			100	ns
dfdt	DACK False to DRQ True	30	90		ns
dfrt	DACK False to REQ True (ACK False)			75	ns
dhwr	DMA Data Hold Time from End of WR	10			ns
dkhw	DACK Hold from End of WR	0			ns
dsrt	SCSI Data Setup to REQ True	25			ns
dswd	Data Setup to End of DMA Write Enable (no μP Parity) (Note 1) (with μP Parity)	35 35			ns ns
dtdf	DACK True to DRQ False			45	ns
eop	Width of EOP Pulse (Note 2)	25			ns
wwn	DMA Non-block Mode Write Enable Width (Note 2)	40			ns

Note 1: Write enable (DMA) is DACK and WR active.

Note 2: EOP, DACK, RD/WR must all be true for recognition of EOP.

12.0 AC Electrical Characteristics (Continued)

12.6 DMA WRITE (NON-BLOCK MODE) INITIATOR SEND



TL/F/9387–19

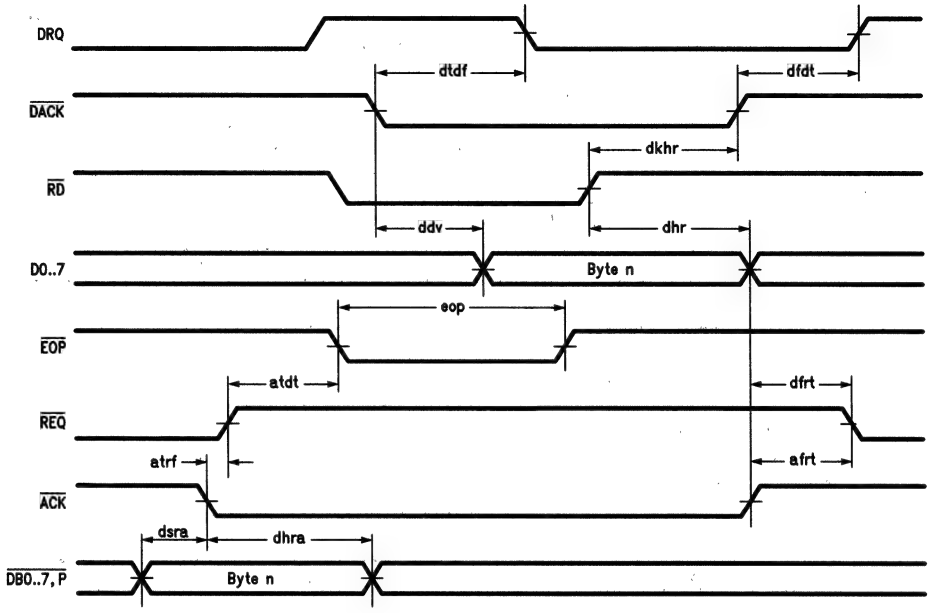
Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
dfaf	DACK False to ACK False			90	ns
dfdt	DACK False to DRQ True	30	90		ns
dhi	SCSI Data Hold from Write Enable-Initiator	15			ns
dhwr	DMA Data Hold Time from End of WR	10			ns
dkhw	DACK Hold from End of WR	0			ns
dswd	Data Setup to End of DMA Write Enable (no μP Parity) (Note 1) (with μP Parity)	35 35			ns ns
dtdf	DACK True to DRQ False			45	ns
eop	Width of EOP Pulse (Note 2)	25			ns
rfdt	REQ False to DRQ True			60	ns
rtat	REQ True to ACK True			80	ns
wwn	DMA Write Enable Width (Note 1)	40			ns

Note 1: Write enable (DMA) is DACK and WR active.

Note 2: EOP, DACK, RD/WR must all be true for recognition of EOP.

12.0 AC Electrical Characteristics (Continued)

12.7 DMA READ (NON-BLOCK MODE) TARGET RECEIVE



TL/F/9387–20

Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
afrt	ACK False to REQ True (DACK or WR False)			75	ns
atdt	ACK True to DRQ True			55	ns
atrf	ACK True to REQ False			100	ns
ddv	DMA Data Valid from Read Enable (Note 1)			40	ns
dfdt	DACK False to DRQ True	30	90		ns
dfrt	DACK False to REQ True (ACK False)			75	ns
dhr	Data Hold from End of Read Enable (Notes 1, 2)	20		60	ns
dhra	SCSI Data Hold from ACK True	15			ns
dkhr	DACK Hold from End of RD	0			ns
dsra	SCSI Data Setup Time to ACK True	10			ns
dtdf	DACK True to DRQ False			45	ns
eop	Width of EOP Pulse (Note 3)	25			ns

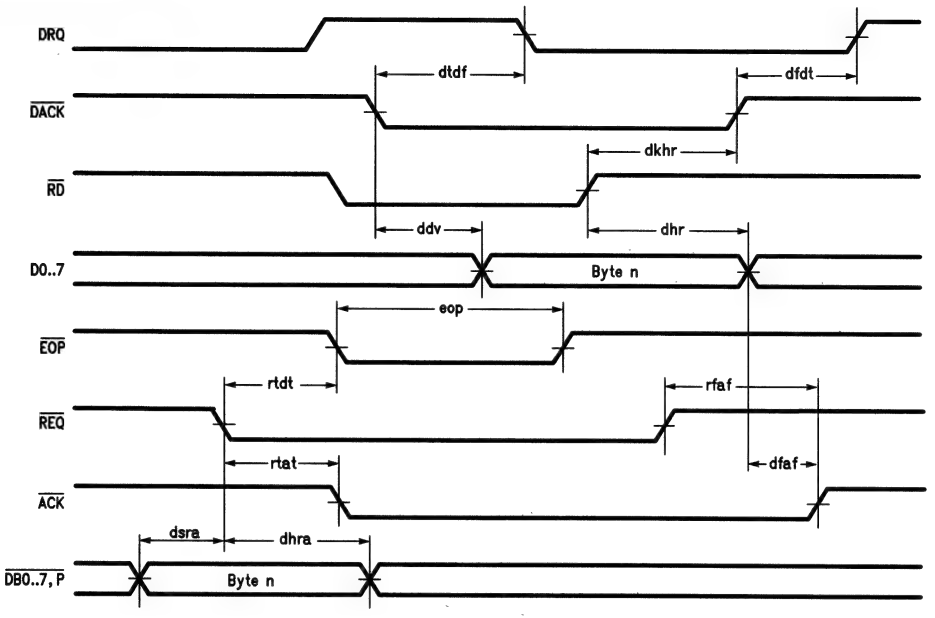
Note 1: Read enable (DMA) is DACK and RD active.

Note 2: This includes the RC delay inherent in the test's method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

Note 3: EOP, DACK, RD/WR must all be true for recognition of EOP.

12.0 AC Electrical Characteristics (Continued)

12.8 DMA READ (NON-BLOCK MODE) INITIATOR RECEIVE



TL/F/9387–21

Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
ddv	DMA Data Valid from Read Enable (Note 1)			40	ns
dfaf	DACK False to ACK False (REQ False)			90	ns
dfdt	DACK False to DRQ True	30	90		ns
dhr	Data Hold from End of Read Enable (Notes 1, 2)	20		60	ns
dhra	SCSI Data Hold from REQ True	15			ns
dkhr	DACK Hold from End of RD	0			ns
dsra	SCSI Data Setup Time to REQ True	10			ns
dtdf	DACK True to DRQ False			45	ns
eop	Width of EOP Pulse (Note 3)	25			ns
rfaf	REQ False to ACK False (DACK False)			90	ns
rtat	REQ True to ACK True			80	ns
rtdt	REQ True to DRQ True			70	ns

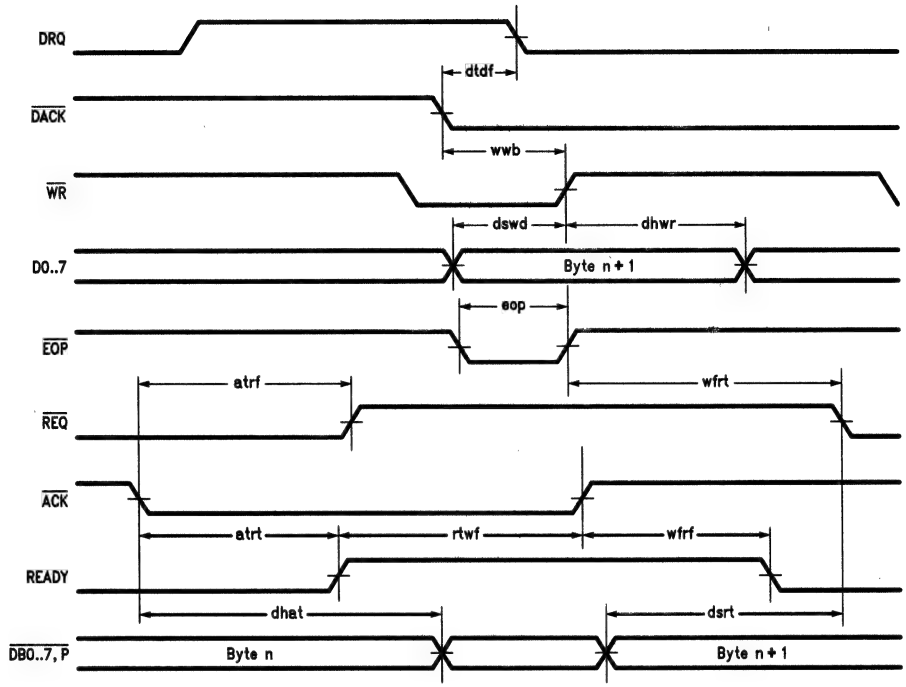
Note 1: Read enable (DMA) is DACK and RD active.

Note 2: This includes the RC delay inherent in the test's method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

Note 3: EOP, DACK, RD/WR must all be true for recognition of EOP.

12.0 AC Electrical Characteristics (Continued)

12.9 DMA WRITE (BLOCK MODE) TARGET SEND



TL/F/9387–22

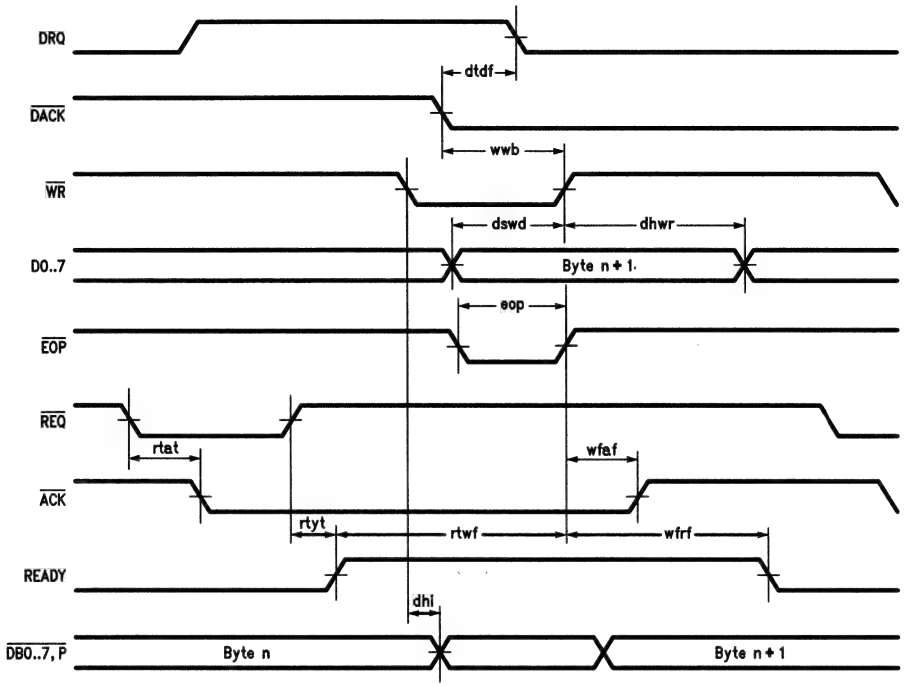
Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
afrt	ACK False to REQ True (DACK or WR False)			75	ns
atrf	ACK True to REQ False			100	ns
atrt	ACK True to READY True			50	ns
dhat	SCSI Data Hold from ACK True	40			ns
dhwr	DMA Data Hold Time from End of WR	10			ns
dsrt	SCSI Data Setup to REQ True	35			ns
dswd	Data Setup to End of DMA Write Enable (no μP Parity) (Note 1) (with μP Parity)	35 35			ns ns
dtdf	DACK True to DRQ False			45	ns
eop	Width of EOP Pulse (Note 2)	25			ns
rtwf	READY True to WR False	40			ns
wfrf	WR False to READY False			50	ns
wfrt	WR False to REQ True (ACK False)			80	ns
wwb	DMA Write Enable Width (Note 1)	40			ns

Note 1: Write enable (DMA) is DACK and WR active.

Note 2: EOP, DACK, RD/WR must all be true for recognition of EOP.

12.0 AC Electrical Characteristics (Continued)

12.10 DMA WRITE (BLOCK MODE) INITIATOR SEND



TL/F/9387–23

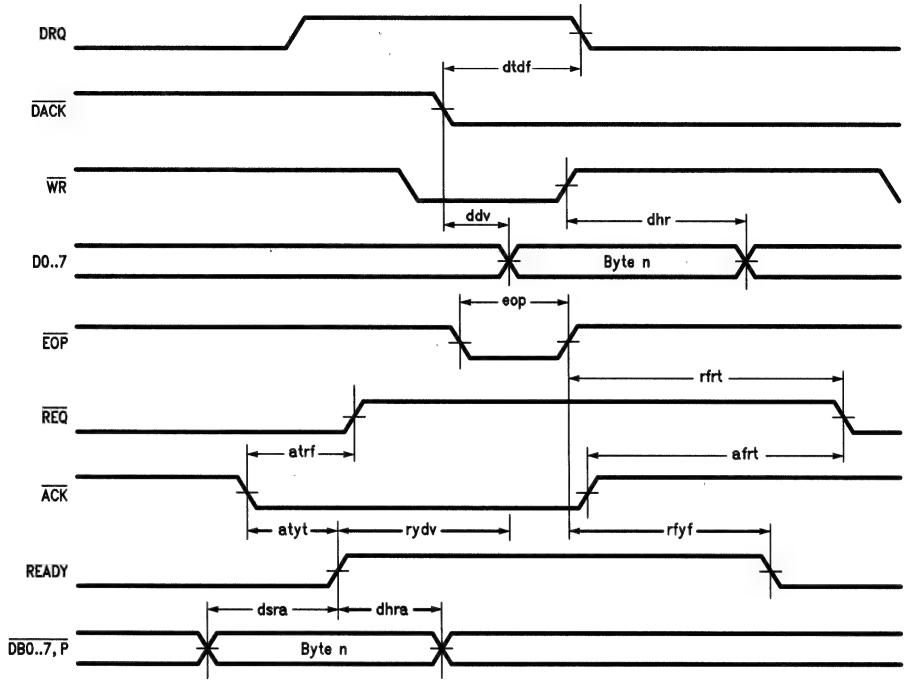
Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
dhi	SCSI Data Hold from Write Enable	15			ns
dhwr	DMA Data Hold Time from End of WR	10			ns
dswd	Data Setup to End of DMA Write Enable (no μP Parity)	35			ns
	(Note 1) (with μP Parity)	35			ns
dtdf	DACK True to DRQ False			45	ns
eop	Width of EOP Pulse (Note 2)	25			ns
rfyt	REQ False to READY True			60	ns
rtat	REQ True to ACK True			80	ns
rtwf	READY True to WR False	40			ns
wfaf	WR False to ACK False (REQ False)			95	ns
wfrf	WR False to READY False			50	ns
wwb	DMA Write Enable Width (Note 1)	40			ns

Note 1: Write enable (DMA) is DACK and WR active.

Note 2: EOP, DACK, RD/WR must all be true for recognition of EOP.

12.0 AC Electrical Characteristics (Continued)

12.11 DMA WRITE (BLOCK MODE) TARGET RECEIVE



TL/F/9387–24

Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
afrt	ACK False to REQ True (DACK or WR False)			75	ns
atrf	ACK True to REQ False			100	ns
atyt	ACK True to READY True			50	ns
ddv	DMA Data Valid from Read Enable (Note 1)			40	ns
dhr	Data Hold from End of Read Enable (Notes 1,2)	20		60	ns
dhra	SCSI Data Hold from ACK True	15			ns
dsra	SCSI Data Setup Time to ACK True	10			ns
dtdf	DACK True to DRQ False			45	ns
eop	Width of EOP Pulse (Note 3)	25			ns
rfrt	RD False to REQ True (ACK False)			75	ns
rfyf	RD False to READY False			45	ns
rydv	READY True to Data Valid			20	ns

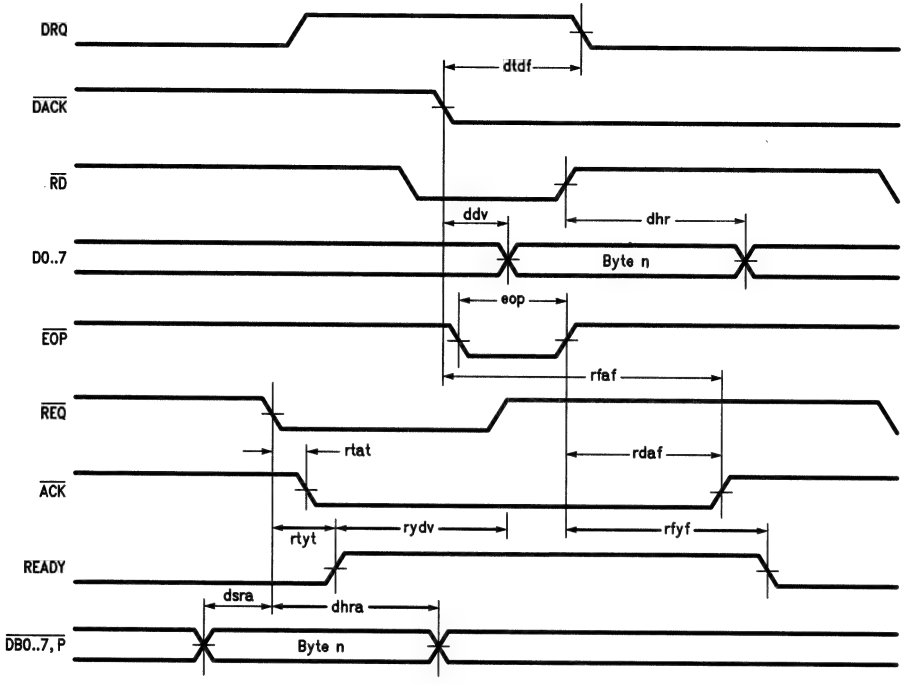
Note 1: Read enable (DMA) is DACK and RD active.

Note 2: This includes the RC delay inherent in the test's method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

Note 3: EOP, DACK, RD/WR must all be active for recognition of EOP.

12.0 AC Electrical Characteristics (Continued)

12.12 DMA READ (BLOCK MODE) INITIATOR RECEIVE



TL/F/9387–25

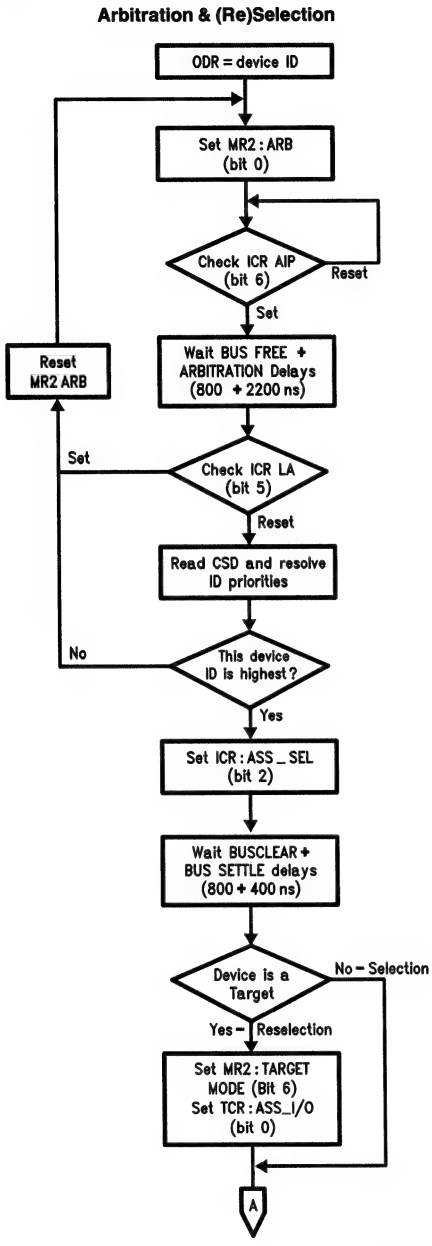
Symbol	Parameter	DP8490			Units
		Min	Typ	Max	
ddv	DMA Data Valid from Read Enable (Note 1)			40	ns
dhr	Data Hold from End of Read Enable (Notes 1,2)	20		60	ns
dhra	SCSI Data Hold from REQ True	15			ns
dsra	SCSI Data Setup Time to REQ True	10			ns
dtdf	DACK True to DRQ False			45	ns
eop	Width of EOP Pulse (Note 3)	25			ns
rdaf	RD False to ACK False (REQ False)			120	ns
rfaf	REQ False to ACK False (DACK False)			90	ns
rfyf	RD False to READY False			45	ns
rtat	REQ True to ACK True			80	ns
rtyt	REQ True to READY True			65	ns
rydv	READY True to Data Valid			20	ns

Note 1: Read enable (DMA) is DACK and RD active.

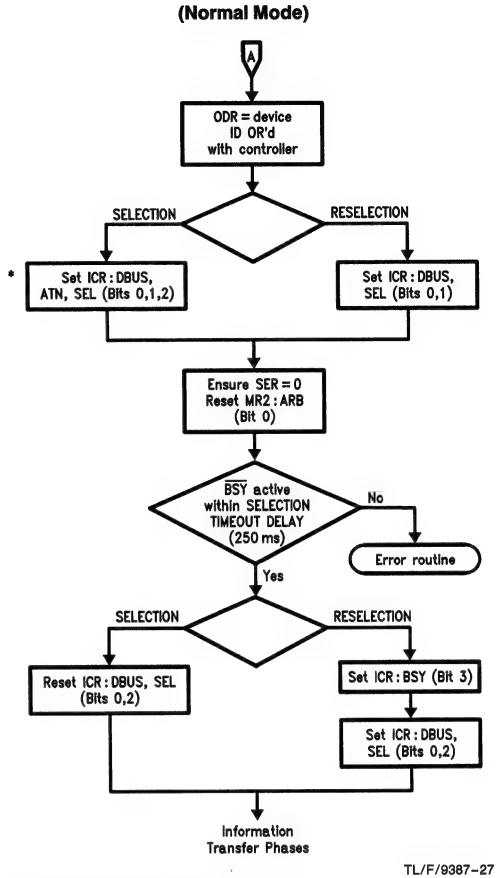
Note 2: This includes the RC delay inherent in the test's method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

Note 3: EOP, DACK, RD/WR must all be active for recognition of EOP.

Appendix A1



TL/F/9387–26

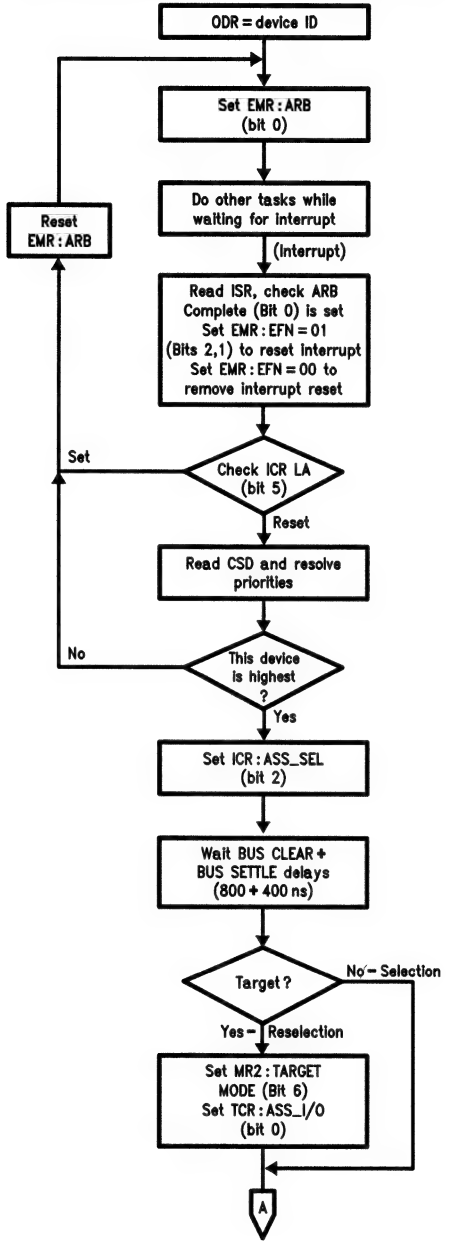


TL/F/9387–27

*Only set ATN if Select with ATN is desired.

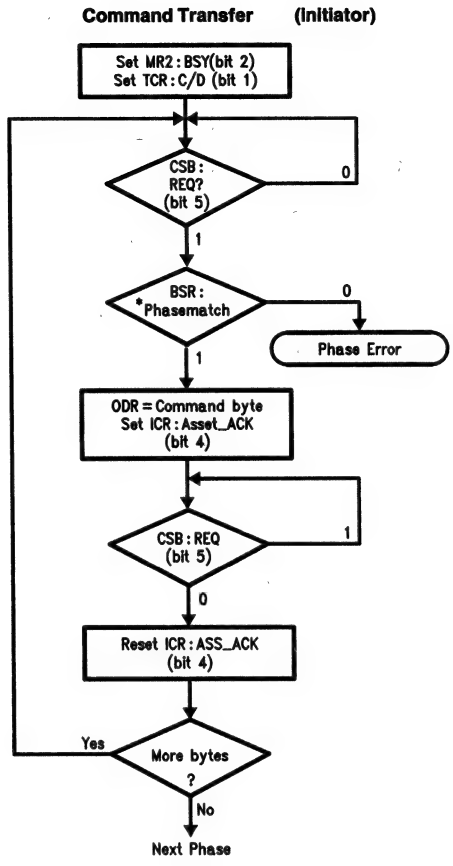
Appendix A1 (Continued)

Arbitration & (Re)Selection (Enhanced Mode)



TL/F/9387–28

Appendix A1 (Continued)

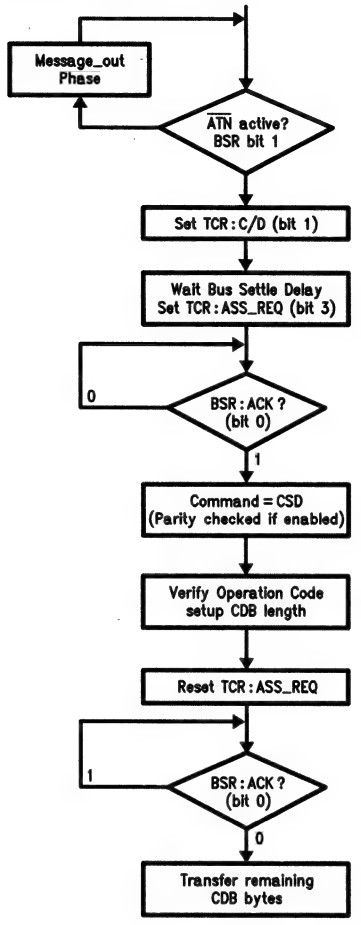


TL/F/9387–29

*This step unnecessary in MODE E if the EMR : APHS (bit 7) is enabled. Logic automatically checks the phase on an error.

Appendix A1 (Continued)

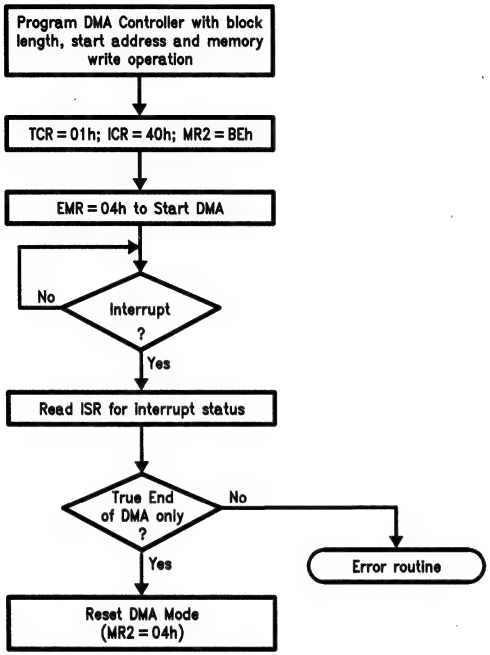
Command Transfer (Target)



TL/F/9387–30

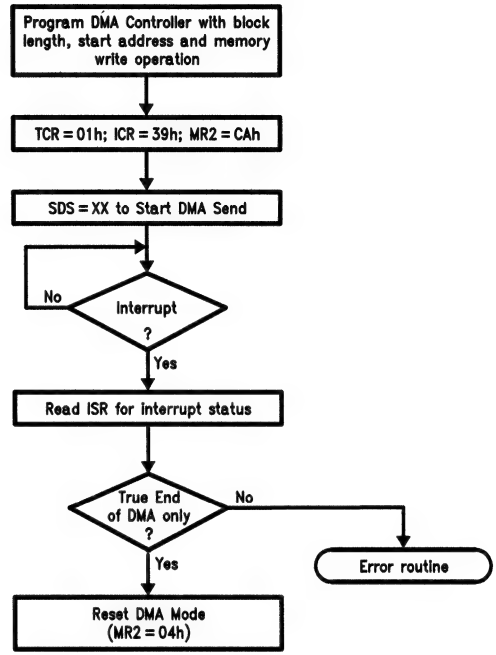
Appendix A1 (Continued)

Block Mode DMA Transfer
Initiator Receive (MODE E)



TL/F/9387–31

Block Mode DMA Transfer
Target Send (MODE E)



TL/F/9387–32

Appendix A2

Register Chart

Read

Current SCSI Data (CSD)

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Initiator Command Register (ICR)

Bit 7							Bit 0
$\overline{RST}$	AIP	LA	$\overline{ACK}$	$\overline{BSY}$	$\overline{SEL}$	$\overline{ATN}$	DBUS

Mode Register 2 (MR2)

Bit 7							Bit 0
BLK	TARG	PCHK	PINT	EOP	BSY	DMA	ARB

Target Command Register (TCR)

Bit 7							Bit 0
0	0	0	0	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$

Current SCSI Bus Status (CSB)

Bit 7							Bit 0
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$	$\overline{SEL}$	$\overline{DBP}$

Bus and Status Register (BSR)

Bit 7							Bit 0
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

Input Data Register (IDR)

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Reset Parity/Interrupt (RPI)—MODE N

Bit 7							Bit 0
X	X	X	X	X	X	X	X

Enhanced Mode Register (EMR)

Bit 7							Bit 0
APHS	MPEN	MPOL	SPOL	LOOP	EFN1	EFN0	ARB

Interrupt Status Register (ISR)

Bit 7							Bit 0
SPE	MPE	EDMA	DPHS	APHS	BSY	SEL	ARB

Target Command Register (TCR)—MODE E

Bit 7							Bit 0
(true) EDMA	0	0	0	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$

X = Unknown

Write

Output Data Register (ODR)

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Initiator Command Register (ICR)

Bit 7							Bit 0
$\overline{RST}$	MODE E	DIFF EN	$\overline{ACK}$	$\overline{BSY}$	$\overline{SEL}$	$\overline{ATN}$	DBUS

Mode Register 2 (MR2)

Bit 7							Bit 0
BLK	TARG	PCHK	PINT	EOP	BSY	DMA	ARB

Target Command Register (TCR)

Bit 7							Bit 0
X	X	X	X	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$

Select Enable Register (SER)

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Start DMA Send (SDS)

Bit 7							Bit 0
X	X	X	X	X	X	X	X

Start DMA Target Receive (SDT)

Bit 7							Bit 0
X	X	X	X	X	X	X	X

Start DMA Initiator Receive (SDI)—MODE N

Bit 7							Bit 0
X	X	X	X	X	X	X	X

Enhanced Mode Register (EMR)

Bit 7							Bit 0
APHS	MPEN	MPOL	SPOL	LOOP	EFN1	EFN0	ARB

Interrupt Mask Register (IMR)

Bit 7							Bit 0
SPE	MPE	EDMA	DPHS	APHS	BSY	SEL	ARB

Target Command Register (TCR)—MODE E

Bit 7							Bit 0
X	X	X	X	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$

X = Don't Care

National Semiconductor

DP5380 Asynchronous SCSI Interface (ASI)

General Description

The DP5380 ASI is a CMOS device designed to provide a low cost, high performance Small Computer Systems Interface. It complies with the ANS X3.131-1986 SCSI standard as defined by the ANSI X3T9.2 committee. It can act as both INITIATOR and TARGET, making it suitable for any application. The ASI supports selection, reselection, arbitration and all other bus phases. High-current open-drain drivers on chip reduce application chip count by interfacing direct to the SCSI bus. An on-chip oscillator provides all timing delays.

The DP5380 is pin and program compatible with the NMOS NCR5380 device. NCR5380 or AM5380 applications can use it with no changes to hardware or software. The DP5380 is available in a 40-pin DIP or a 44-pin PCC.

The ASI is intended to be used in a microprocessor based application, and achieves maximum performance with a DMA controller. The device is controlled by reading and writing several internal registers. A standard non-multiplexed address and data bus easily fits any μP environment.

Data transfers can be performed by programmed-I/O, pseudo-DMA or via a DMA controller. The ASI easily interfaces to a DMA controller using normal or Block Mode. The ASI can be used in either a polled or interrupt-driven environment.

Features

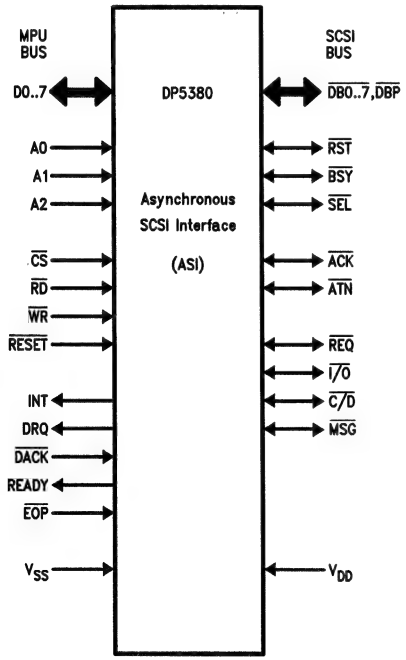
SCSI Interface

- Supports TARGET and INITIATOR roles
- Parity generation with optional checking
- Arbitration support
- Direct control/monitoring of all SCSI signals
- High current outputs drive SCSI bus directly
- Faster and improved timing
- Very low SCSI bus loading

μP Interface

- Memory or I/O-mapped control transfers
- Programmed-I/O or DMA data transfers
- Normal or Block-mode DMA
- Fast DMA handshake timing

Connection Diagram



TL/F/9756–1

Table of Contents

1.0 Functional Description

1.1 OVERVIEW

The ASI is designed to be used as a peripheral device in a μP-based application and appears as a number of read/write registers. Write registers are programmed to select desired functions. Status registers provide indication of operating conditions.

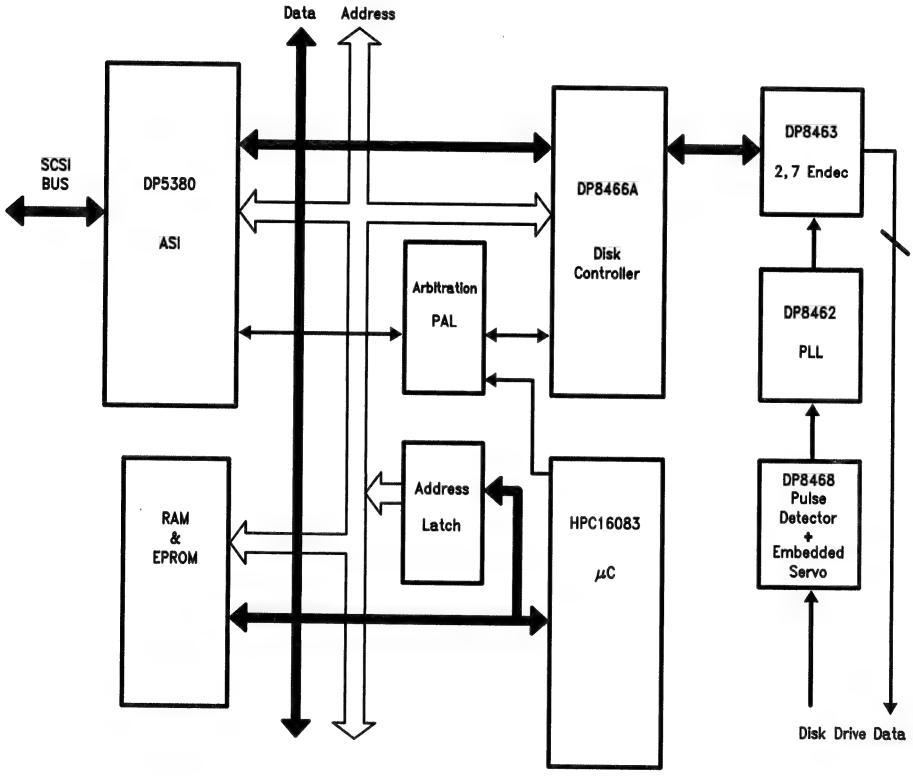
For best performance a DMA controller can be easily interfaced directly to the ASI. The ASI provides request/acknowledge and wait-state signals for the DMA interface.

The SCSI bus is easily controlled via the ASI registers. Any bus signal may be asserted or deasserted via a bit in the appropriate register, and the state of every signal is available by reading registers. This direct control over SCSI signals allows the user to implement all or part of the protocol in firmware. The ASI provides hardware support for much of the protocol.

The ASI provides the following SCSI support:

- Programmed-I/O transfers for all eight information transfer types, with or without parity.
- Data transfers via DMA, in either block or non-block mode. The DMA interface supports most devices.
- Individual setting/resetting and monitoring of every SCSI bus signal.
- Automatic release of the bus for BSY loss from a TARGET, SCSI RST, and lost arbitration.
- Automatic bus arbitration—the μP has only to check for highest priority.
- Selection or Reselection of any bus device. The ASI will respond to both Selection and Reselection.
- Optional automatic monitoring of the $\overline{\text{BSY}}$ signal from a TARGET with an interrupt after releasing control of the bus.

Figure 1 shows an ASI in a typical application, a low cost embedded SCSI disk controller. In this application the 8051 single-chip μP acts as the controller and the dual DMA channels in the DP8475 allow one for the disk data and the other for SCSI data. The PAL® provides chip selection as well as determining who has control of the bus. The advantage of using a μP with on-board ROM is that there is more free time on the external bus.



TL/F/9756–2

FIGURE 1. ASI Application

1.0 Functional Description (Continued)

1.2 μP INTERFACE

Figure 2 shows a block diagram of the ASI. Key blocks within the ASI are Read/Write registers with associated decode and control logic, interrupt and DMA logic, SCSI bus arbitration logic, SCSI drivers/receivers with parity and the SCSI data input and output registers. The ASI has three interfaces, one to SCSI, one to a DMA controller and the third to a μP. The internal registers control all operations of the ASI.

The μP interface consists of non-multiplexed address and data busses with associated control signals. Address decode logic selects a register for reading or writing. The address lines A0–2 select the register for μP accesses while for DMA accesses the address lines are ignored.

The register bank consists of twelve registers mapped into an address space of eight locations. Upon an external chip reset the registers are cleared (all zeroes).

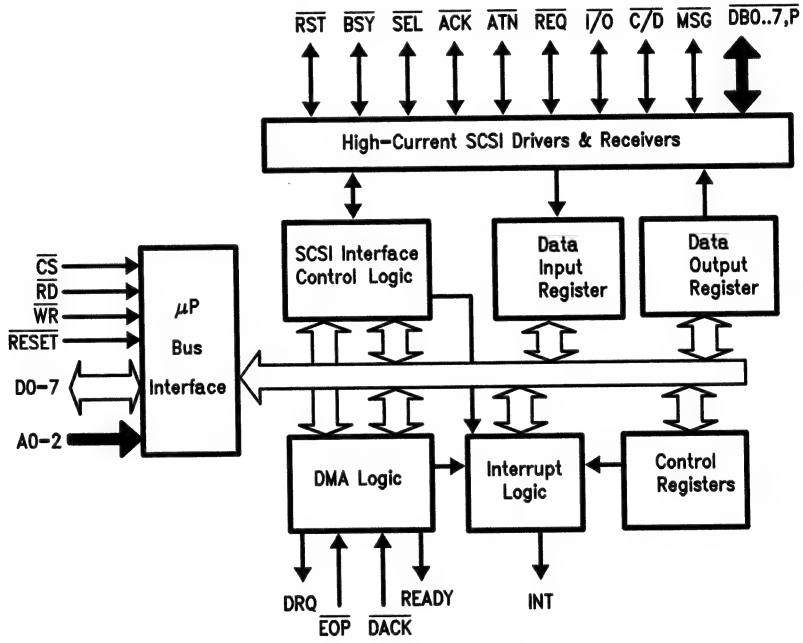
1.3 DMA INTERFACE

The DMA logic interfaces to single-cycle, block mode, flow-through or fly-by controllers. Single byte transfers are accomplished via the DRQ/$\overline{\text{DACK}}$ handshake signals. Block mode transfers use the READY output to control the speed (insert wait-states). An End Of Process ($\overline{\text{EOP}}$) input from the DMA controller signals the ASI to halt DMA transfers. An interrupt can be generated for DMA completion or an error (see Section 5.0). All DMA data passes through the SCSI data input and output registers, automatically selected during DMA cycles.

1.4 SCSI INTERFACE

The ASI contains all logic required to interface directly to the SCSI bus. Direct control and monitoring of all SCSI signals is provided. The state of each SCSI signal may be determined by reading a register which continuously reflects the state of the bus. Each signal may be asserted by writing a ONE to the appropriate bit.

The ASI includes logic to automatically handle SCSI timing sequences too fast for μP control. In particular there is hardware support for DMA transfers, bus arbitration, selection/reselection, bus phase monitoring, $\overline{\text{BSY}}$ monitoring for bus disconnection, bus reset and parity generation and checking.



TL/F/9756–3

FIGURE 2. ASI Block Diagram

1.0 Functional Description (Continued)

The ASI arbitration logic controls arbitration for use of the SCSI bus. The μP programs the SCSI device ID into the ASI, then sets the ARBITRATE bit. The INITIATOR COMMAND REGISTER (ICR) is read to determine when arbitration has started and whether it is won or lost.

The $\overline{BSY}$ signal is continously monitored to detect bus disconnection and bus free phases. The ASI incorporates an on-board oscillator to determine Bus Settle, Bus Free and Arbitration Delays. The oscillator tolerance guarantees all timing to be within the SCSI specification.

The ASI incorporates high-current drivers and SCHMITT trigger receivers for interfacing directly to the SCSI bus. This feature reduces the chip count of any SCSI application.

1.5 PARITY

The ASI provides for parity protection on the SCSI interface. The data bus has eight data bits and one parity bit. The parity may be enabled via a register bit. A parity error can be programmed to cause an interrupt.

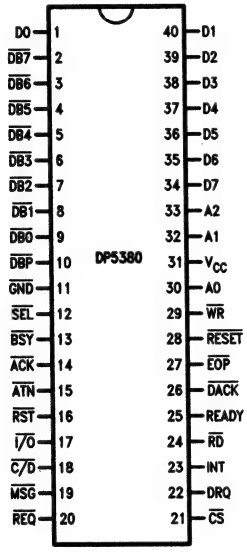
2.0 Pin Descriptions

Symbol	DIP	PCC	Type	Function
$\overline{CS}$	21	24	I	**Chip Select:** an active low enable for read or write operations, accessing the register selected by A0 . . . 2.
A0 . . . 2	30, 32, 33	33, 36, 37	I	**Address 0 . . . 2:** these three signals are used with $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ to address a register for read or write.
$\overline{RD}$	24	27	I	**Read:** an active low enable for reading an internal register selected by A0 . . . 2 and enabled by $\overline{CS}$. It also selects the Input Data Register when used with $\overline{DACK}$.
$\overline{WR}$	29	32	I	**Write:** an active low enable for writing an internal register selected by A0 . . . 2 and enabled by $\overline{CS}$. It also selects the Output Data Register when used with $\overline{DACK}$.
$\overline{RESET}$	28	31	I	**Reset:** an active low input with a Schmitt trigger. Clears all internal registers. (SCSI $\overline{RST}$ unaffected).
D0 . . . 7	1, 40–34	2, 44–38	I/O	**Data 0 . . . 7:** bidirectional TRI-STATE® signals connecting the active high μP data bus to the internal registers.
INT	23	26	O	**Interrupt:** an active high output to the μP when an error has occurred, an event requires service or has completed.
$\overline{DRQ}$	22	25	O	**DMA Request:** an active high output asserted when the data register is ready to read or written. DRQ occurs only if DMA mode is enabled. The signal is cleared by $\overline{DACK}$.
$\overline{DACK}$	26	29	I	**DMA Acknowledge:** an active low input that resets DRQ and addresses the data registers for input or output transfers. $\overline{DACK}$ is used instead of $\overline{CS}$ by the DMA controller.
READY	25	28	O	**Ready:** an active high output used to control the speed of block mode DMA transfers. Ready goes active when the chip is ready to send/receive data and remains inactive after the transfer until the byte is sent or until the DMA mode bit is reset.
$\overline{EOP}$	27	30	I	**End Of Process:** an active low signal that terminates a block of DMA transfers. It should be asserted during the transfer of the last byte.
$\overline{DB0}$ . . . 7 $\overline{DBP}$	9 . . . 2, 10	10 . . . 3, 11	I/O	**DB0 . . . 7, DBP:** SCSI data bus with parity. $\overline{DB7}$ is the MSB and is the highest priority during arbitration. Parity is ODD. Parity is always generated and can be optionally checked. Parity is not valid during arbitration.
$\overline{RST}$	16	18	I/O	**Reset:** SCSI reset, monitored and can be set by ASI.
$\overline{BSY}$	13	15	I/O	**Busy:** indicates the SCSI bus is being used. Can be driven by TARGET or INITIATOR.
$\overline{SEL}$	12	14	I/O	**Select:** used by the INITIATOR to select a TARGET or by the TARGET to reselect an INITIATOR.
$\overline{ACK}$	14	16	I/O	**Acknowledge:** driven by the INITIATOR and received by the TARGET as part of the REQ/$\overline{ACK}$ handshake.
$\overline{ATN}$	15	17	I/O	**Attention:** driven by the INITIATOR to indicate an attention condition to the TARGET.

2.0 Pin Descriptions (Continued)

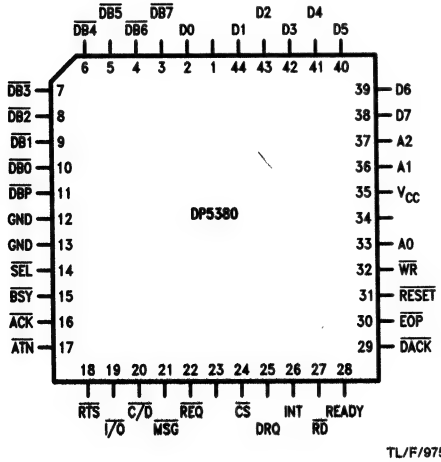
Symbol	DIP	PCC	Type	Function
$\overline{REQ}$	20	22	I/O	**Request:** driven by the TARGET and received by the INITIATOR as part of the $\overline{REQ}$/$\overline{ACK}$ handshake.
$\overline{I/O}$	17	19	I/O	**Input/Output:** driven by the TARGET to control the direction of transfers on the SCSI bus. This signal also distinguishes between selection and reselection.
$\overline{C/D}$	18	20	I/O	**Command/Data:** driven by the TARGET to indicate whether command or data bytes are being transferred.
$\overline{MSG}$	19	21	I/O	**Message:** driven by the TARGET during message phase to identify message bytes on the bus.
VCC GND	31 11	35 12, 13	—	**VCC, GND:** +5V DC is required. Because of very large switching currents good decoupling and power distribution is mandatory.

2.1 Connection Diagrams



TL/F/9756–4

Order Number DP5380N
See NS Package Number N40A



TL/F/9756–5

Order Number DP5380V
See NS Package Number V44A

3.0 Register Description

3.1 GENERAL

The DP5380 ASI is a register-based device with eight addressable locations. Some addresses have dual functions depending upon whether they are being read from or written to. Device operation is described in Section 4.

Figure 3.2 summarises the register map. Note that for registers reading or writing SCSI signals the SCSI name is used for each bit. Although the SCSI bus is active low the registers invert the SCSI bus. This means an active SCSI signal is represented by a ONE in a register and an inactive signal by a ZERO.

3.2 REGISTERS

OUTPUT DATA REGISTER (ODR)

8 Bits HA 0 Write-Only

This is a transparent latch used to send data to the SCSI bus. The register can be written by µP cycles or via DMA. DMA writes automatically select the ODR at Hex Address 0 (HA 0). This register is also written with the ID bits required during arbitration and selection/reselection phases. Data is latched at the end of the write cycle.

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Output Data Register

CURRENT SCSI DATA (CSD)

8 Bits HA 0 Read-Only

This register enables reading of the current SCSI data bus. If SCSI parity checking is enabled it will be checked at the beginning of the read cycle. The register is also used for µP accesses of SCSI data during programmed-I/O or ID checking during arbitration. Parity is not valid during arbitration. DMA transfers select the IDR (HA 6) instead of the CSD register.

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Current SCSI Data

INITIATOR COMMAND REGISTER (ICR)

8 Bits HA 1 Read-Write

This register is used to control the INITIATOR and some other SCSI signals, and to monitor the progress of bus arbitration. Most of the SCSI signals may also be asserted in TARGET mode. Bits 5 to 0 are reset when $\overline{BSY}$ is lost (see MR2 description).

Bit 7							Bit 0
$\overline{RST}$	TEST	LA/DIFF	$\overline{ACK}$	$\overline{BSY}$	$\overline{SEL}$	$\overline{ATN}$	DBUS

Initiator Command Register

DBUS: Assert Data Bus **Bit 0**

0 Disable SCSI data bus driving.

1 Enable contents of Output Data Register onto the SCSI data bus. SCSI parity is also generated and driven on DBP.

This bit should be set when transferring data out of the ASI in either TARGET or INITIATOR mode, for both DMA or programmed-I/O. In INITIATOR mode the drivers are only enabled if: Mode Register 2 TARGET MODE bit is 0, and $\overline{I/O}$ is false, and $\overline{C/D}$, $\overline{I/O}$, $\overline{MSG}$ match the contents of the Target Command Register (phasematch is true). In TARGET mode only the MR2 bit needs to be set with this bit.

Reading the ICR reflects the state of this bit.

$\overline{ATN}$: Assert Attention **Bit 1**

0 Deassert $\overline{ATN}$.

1 Assert SCSI $\overline{ATN}$ signal. The MR2 TARGET MODE bit must also be false to assert the signal.

Reading the ICR reflects the state of this bit.

$\overline{SEL}$: Assert Select **Bit 2**

0 Deassert $\overline{SEL}$.

1 Assert SCSI $\overline{SEL}$ signal. Can be used in INITIATOR or TARGET mode.

Reading the ICR reflects the state of this bit.

$\overline{BSY}$: Assert Busy **Bit 3**

0 Deassert $\overline{BSY}$.

1 Assert SCSI $\overline{BSY}$ signal. Can be used in INITIATOR or TARGET mode.

Reading the ICR reflects the state of this bit.

Hex Adr	Register	Mnemonic	Bits	R/W
0	Output Data Register	ODR	8	WO
0	Current SCSI Data	CSD	8	RO
1	Initator Command Register	ICR	8	RW
2	Mode Register 2	MR2	8	RW
3	Target Command Register	TCR	4	RW
4	Select Enable Register	SER	8	WO
4	Current SCSI Bus Status	CSB	8	RO
5	Bus and Status	BSR	8	RO
5	Start DMA Send	SDS	0	WO
6	Start DMA Target Receive	SDT	0	WO
6	Input Data Register	IDR	8	RO
7	Start DMA Initiator Receive	SDI	0	WO
7	Reset Parity/Interrupts	RPI	0	RO

FIGURE 3.2. Registers

3.0 Register Description (Continued)

$\overline{ACK}$: Assert Acknowledge **Bit 4**

0 Deassert $\overline{ACK}$.

1 Assert SCSI $\overline{ACK}$ signal. The MR2 TARGET MODE bit must also be false to assert the signal.

Reading the ICR reflects the state of this bit.

DIFF: Differential Enable **Bit 5 Write**

0 This bit must be reset to 0.

1 Do not use. Reserved for future use on a differential pair device.

LA: Lost Arbitration **Bit 5 Read**

0 Normally reset to 0 to show arbitration not lost or not enabled.

1 Will be set when the ASI loses arbitation, i.e. when SEL is true during arbitration AND the Assert $\overline{SEL}$ bit of this register is false.

A 1 in this bit means the ASI has arbitrated for the bus, asserted $\overline{BSY}$ and its ID on the data bus and another device has asserted $\overline{SEL}$. The ARBITRATE bit in MR2 must be set to enable arbitration.

TEST: Test Mode Enable **Bit 6 Write**

0 Output drivers are enabled.

1 Output drivers disabled.

AIP: Arbitration In Progress **Bit 6 Read**

0 Normally 0 to show no arbitration in progress.

1 Set when the ASI has detected BUS FREE phase and asserted $\overline{BSY}$ and the Output Data Register contents onto the SCSI data bus. This bit remains set until arbitration is disabled.

$\overline{RST}$: Assert $\overline{RST}$ **Bit 7**

0 Deassert $\overline{RST}$.

1 Assert SCSI $\overline{RST}$ signal. $\overline{RST}$ is asserted as long as this bit is 1, or until a μP Reset ($\overline{RESET}$).

After this bit is set the INT pin goes active and internal registers reset (except for the interrupt latch, MR2 TARGET MODE bit, and this bit. Reading the ICR reflects the state of this bit.

MODE REGISTER 2 (MR2)
8 Bits HA2 Read-Write

This register is used to program basic operating conditions in the ASI. Operation as TARGET or INITIATOR, DMA mode and type as well as some interrupt controls are set via this register. This is a Read/Write register and when read the value reflects the state of each bit.

Bit 7							Bit 0
BLK	TARG	PCHK	PINT	EOP	BSY	DMA	ARB

Mode Register 2

ARB: Arbitrate **Bit 0**

0 Disable arbitration.

1 Enable arbitration. The ASI will wait for a BUS FREE phase then arbitrate for the bus. Before setting this bit the Output Data Register should contain the SCSI device ID—a single bit set only. The status of the arbitration process is given in the AIP and LA bits (6, 5) in the Initiator Command Register.

DMA: DMA Mode **Bit 1**

0 Disable DMA mode.

1 Enable DMA operation. This bit should be set then one of address 5 to 7 written to start DMA. The TARGET MODE bit in the ICR and the phase lines in the TCR should have been set appropriately. The DBUS bit in the ICR must be set for DMA operations. $\overline{BSY}$ must be active in order to set this bit. The phase lines must match the contents of the TCR during the actual transfers. In DMA mode ASI logic automatically controls the $\overline{REQ}$/$\overline{ACK}$ handshakes.

This bit should be reset by a μP write to stop any DMA transfer. An $\overline{EOP}$ signal will not reset this bit. During DMA, $\overline{CS}$ and $\overline{DACK}$ should not be active simultaneously.

This bit will be reset if $\overline{BSY}$ is lost during DMA mode.

BSY: Monitor Busy **Bit 2**

0 Disable $\overline{BSY}$ monitor.

1 Monitor SCSI $\overline{BSY}$ signal and interrupt when $\overline{BSY}$ goes inactive. When this bit goes active the lower 6 bits of the ICR are reset and all signals removed from the SCSI bus. This is used to check for valid TARGET connection.

EOP: Enable $\overline{EOP}$ Interrupt **Bit 3**

0 No interrupt for $\overline{EOP}$.

1 Interrupt after valid $\overline{EOP}$ condition.

PINT: Enable SCSI Parity Interrupt **Bit 4**

0 No interrupt on SCSI parity error.

1 When SCSI parity is enabled via the PCHK bit, setting this bit enables an interrupt upon a SCSI parity error.

PCHK: Enable SCSI Parity Checking **Bit 5**

0 No SCSI parity checking.

1 Enable checking of SCSI parity during read operations. This applies to either programmed I/O or DMA mode.

TARG: Target Mode **Bit 6**

0 Initiator Mode.

1 Target Mode.

BLK: Block Mode DMA **Bit 7**

0 Non-block DMA.

1 When set along with DMA bit (1) enable block mode DMA transfers. In block mode the READY line is used to handshake each byte with the DMA controller instead of the DRQ/$\overline{DACK}$ handshake used in non-block mode.

TARGET COMMAND REGISTER (TCR)
4 Bits HA 3 Read-Write

This register is used to control TARGET SCSI signals and to program the desired phase during INITIATOR mode. During

3.0 Register Description (Continued)

DMA transfers the SCSI phase lines (C/D, MSG, I/O) must match the contents of the TCR for transfers to occur. A phase mismatch halts DMA transfers and generates an interrupt.

Bit 7							Bit 0
x	x	x	x	REQ	MSG	C/D	I/O

Target Command Register

This is a read/write register and the value read reflects the state of each bit, except bit 4–7 which always read 0.

I/O: Assert I/O **Bit 0**

0 Deassert I/O.

1 Assert SCSI I/O signal. The MR2 TARGET MODE bit must also be active.

C/D: Assert C/D **Bit 1**

0 Deassert C/D.

1 Assert SCSI C/D signal. The MR2 TARGET MODE bit must also be active.

MSG: Assert MSG **Bit 2**

0 Deassert MSG.

1 Assert SCSI MSG signal. The MR2 TARGET MODE bit must also be active.

REQ: Assert REQ **Bit 3**

0 Deassert REQ.

1 Assert SCSI REQ signal. The MR2 TARGET MODE bit must also be active. This bit is used to handshake SCSI data via programmed-I/O.

SELECT ENABLE REGISTER (SER)
8 Bits HA 4 Write-Only

This write-only register is used to program the SCSI device ID for the ASI to respond to during Selection or Reselection Phases. Only one bit in the register should be set. When SEL is true, BSY false and the SER ID bit active an interrupt will occur.

This interrupt is reset or can be disabled by writing zero to this register. Parity will also be checked during Selection or Reselection if the PCHK bit in MR2 is set.

Bit 7							Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Select Enable Register

CURRENT SCSI BUS STATUS (CSB)
8 Bits HA 4 Read-Only

This read-only register is used to monitor SCSI control signals and the SCSI parity bit. The SCSI lines are monitored during programmed-I/O transfers and after an interrupt in order to determine the cause. A bit is 1 if the corresponding SCSI signal is active.

Bit 7							Bit 0
RST	BSY	REQ	MSG	C/D	I/O	SEL	DBP

Current SCSI Bus Status

BUS AND STATUS REGISTER (BSR)
8 Bits HA 5 Read-Only

This read-only register is used to monitor SCSI signals not included in the CSB, and internal status bits. This register is read after an interrupt to determine the cause of an interrupt. Bit 0 or 1 are set to 1 if the SCSI signal is active.

Bit 7							Bit 0
EDMA	DRQ	SPER	INT	PHSM	BSY	ATN	ACK

Bus & Status Register

ACK: Acknowledge **Bit 0**

This bit reflects the state of the SCSI ACK Signal.

ATN: Attention **Bit 1**

This bit reflects the state of the SCSI ATN Signal.

BSY: Busy Error **Bit 2**

0 No Error.

1 This SCSI BSY signal has become inactive while the MR2 BSY (Monitor BSY) bit is set. This will cause an interrupt, remove all ASI signals from the SCSI bus and reset the DMA MODE bit in MR2.

PHSM: Phase Match **Bit 3**

0 Phase Match. The SCSI C/D, I/O and MSG phase lines are continuously compared with the corresponding bits in the TCR. The result of this comparison is reflected in this bit. This bit must be 1 (phase matches) for DMA transfers. A phase mismatch will stop DMA transfers and cause an interrupt.

INT: Interrupt Request **Bit 4**

0 No Interrupt.

1 Interrupt request active. Set when an enabled interrupt condition occurs. This bit reflects the state of the INT pin. INT may be reset by performing a Reset Parity/Interrupt (RPI) function.

SPER: SCSI Parity Error **Bit 5**

0 No SCSI parity error.

1 SCSI parity error occurred. This bit remains set once an error occurs until the RPI function clears it. The PCHK bit in MR2 must be set for a parity error to be checked and registered.

DRQ: DMA Request **Bit 6**

0 No DMA request.

1 DMA request active. This bit reflects the state of the DRQ pin. DRQ is reset by asserting DACK during a DMA cycle or by resetting the DMA bit in MR2. A Busy error will reset the MR2 DMA bit and thus will also clear DRQ. A phase mismatch will not reset DRQ.

EDMA: End of DMA **Bit 7**

0 Not end of DMA.

1 Set when DACK, EOP and either RD or WR are active simultaneously. Normally occurs when the last byte is transferred by the DMA. During DMA send operations the last byte transferred by the DMA may not have been transferred on SCSI so REQ and ACK should be monitored to verify when the last SCSI transfer is complete. This bit is reset when the MR2 DMA bit is reset.

3.0 Register Description (Continued)

START DMA SEND (SDS)
0 Bits HA 5 Write-Only

This write-only register is used to start a DMA send operation. A write of don't-care data should be the last thing done by the μP. The MR2 DMA, BLK and TARG bits must have been programmed previously.

Bit 7							Bit 0
x	x	x	x	x	x	x	x

Start DMA Send

START DMA TARGET RECEIVE (SDT)
0 Bits HA 6 Write-Only

This write-only register is used to start a DMA Target Receive operation. Same comments as SDS apply.

INPUT DATA REGISTER (IDR)
8 Bits HA 6 Read-Only

This read-only register contains the SCSI data last latched during a DMA receive. Each byte from SCSI is latched into this register automatically by the ASI DMA logic. A DMA read ($\overline{DACK}$ and $\overline{RD}$) automatically selects this register. Programmed-I/O SCSI data reads should use the CSD (HA8)

START DMA INITIATOR RECEIVE (SDI)
0 Bits HA 7 Write-Only

This write-only register is used to start a DMA INITIATOR Receive Operation. Same comments as SDS apply.

RESET PARITY/INTERRUPT (RPI)
0 Bits HA 7 Read-Only

This read-only register is used to reset the parity and interrupt latches. Reading this register resets the SCSI parity, Busy Loss and Interrupt Request latches.

4.0 Device Operation

4.1 GENERAL

This section describes overall operation of the ASI. More detailed information of data transfers, interrupts and reset conditions are covered in later sections. The operation description covers μP accesses, SCSI bus monitoring, arbitration, selection, reselection, programmed-I/O, DMA interrupts. Programming and timing details are covered.

For information regarding interfacing to μP's and DMA controllers refer to Section 7.0.

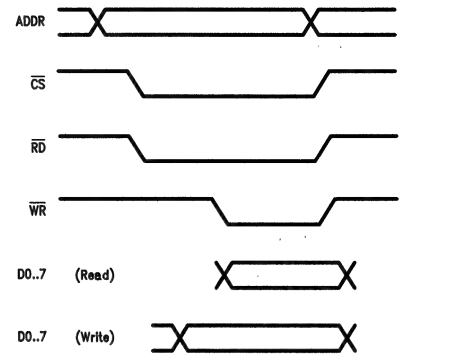
In the descriptions following program examples are given in pseudo-C. This processor-independent approach should be clearest. These are backed up by flow charts in Appendix A.1.

4.2 μP ACCESSES

The μP accesses the EASI via the $\overline{CS}$, $\overline{RD}$, $\overline{WR}$ and address and data lines in order to read/write the registers. *Figure 4.2* shows typical timing. Note the use of non-multiplexed address and data lines.

4.3 SCSI BUS MONITORING/DRIVING

The SCSI bus may be monitored or driven at any time. Each bus signal is buffered and inverted by the ASI and can be read via the CSB, BSR and CSD registers. An active SCSI reads a 1 in the status registers.



TL/F/9756–6

FIGURE 4.2. μP Cycles

Each SCSI signal may be asserted by setting a bit in the TCR or ICR. Setting the bit to 1 asserts the SCSI signal.

The following code demonstrates a byte transferred via programmed-I/O in INITIATOR mode.

```
{     /*Transfer one byte as Initiator*/
   while (NOT (TCR:REQ));
      /* wait till TARGET asserts REQ */
   data = input (CSD);
      /* parity is checked if enabled*/
   output (ICR, Assert ACK);
   while (TCR:REQ);
      /* wait till TARGET deasserts REQ */
   output (ICR, 0);
      /* deassert ACK, ready for next byte */
}
```

4.4 ARBITRATION

This sub-section describes the arbitration support provided by the ASI and how to program it.

Since the SCSI arbitration process requires signal sequencing too fast for μP's, hardware support is provided by the ASI. The arbitration process is enabled by bit 0 MR2 (ARB). Prior to setting this bit the ODR should be programmed with the device's SCSI ID—a single bit.

The ASI will monitor the bus for a BUS FREE phase. The $\overline{BSY}$ signal is continuously monitored. If continuously inactive for at least a SCSI Bus Settle Delay (400 ns) and $\overline{SEL}$ is inactive, a valid Bus Free Phase exists. After a period of SCSI Bus Free Delay (800 ns) the ASI asserts $\overline{BSY}$ and the ODR onto the SCSI data bus. The μP should poll the ICR to determine when arbitration has started. The AIP bit in the ICR is set when the Bus Free Phase is detected and the EASI is beginning the Bus Free Delay. Following the Bus Free Delay a 2.2 μs SCSI Arbitration Delay is required before examining the data bus to resolve the priorities of the ID bits. This delay must be implemented in firmware. The ICR Lost Arbitration (LA) bit must be examined to determine whether arbitration is lost. The LA bit is set if another

4.0 Device Operation (Continued)

device asserts $\overline{SEL}$ during arbitration. If the LA bit is 0 the data bus is read via the CSD register. The data is examined to resolve ID priorities. If this device is the highest ID assert $\overline{SEL}$ by setting ICR bit 2 to a 1. After waiting Bus Clear + Bus Settle Delays (1200 ns) the Selection Phase begins. These 2 delays must be implemented in firmware.

4.5 SELECTION/RESELECTION

The ASI can be used to select or reselect a device. The ASI will also respond to selection or reselection.

4.5.1 Selecting/Reselecting

Selection requires programming the ODR with the desired and own device ID's; the data bus via ICR DBUS (bit 0); asserting $\overline{ATN}$ if required via ICR bit 1; asserting $\overline{SEL}$ via ICR bit 2; then resetting the MR2 ARB bit.

The SER should have been cleared to zero before Selection/Reselection to ensure the ASI does not respond. If Reselection is desired the $\overline{I/O}$ line should also be asserted before $\overline{SEL}$ via TCR bit 0.

Resetting the ARB bit causes the ASI to remove $\overline{BSY}$ and the ODR from the data bus. Thus the ICR Assert data bus bit is required to assert the bits for desired and own device ID's.

$\overline{BSY}$ is then monitored to determine when the device has responded to (re)selection. If the device fails to respond an error handler should sequence the ASI off the bus. If the device responds the ICR DBUS and $\overline{SEL}$ bits should be reset to remove these signals. If this is a Reselection the ICR $\overline{BSY}$ bit (3) should be set before removing the other signals.

The bus is now ready to handle Information Transfer Phases.

4.5.2 (Re)Selection Response

The ASI responds to Selection or Reselection when the SER is non-zero. A (re)selected interrupt is generated when $\overline{BSY}$ is false for at least a Bus Settle Delay (400 ns); and $\overline{SEL}$ is true AND any non-zero bit in the SER has its corresponding SCSI data bus bit active. A Selection is disabled by zeroing the SER. If parity is supported it should be valid during (re)selection so must be checked via the SPE bit (5) in the BSR. SCSI specification states that (re)selection is not valid if more than 2 data bits are active. This condition is checked by reading the CSD.

When the selection interrupt occurs it is determined by reading the BSR and CSB registers. There is no dedicated status bit for (re)selection so it must be determined by the absence of other interrupts, and the active state of the $\overline{SEL}$ signal. Reselection occurs when $\overline{I/O}$ is also active. See Section 6.0.

4.6 MONITORING BSY

While an INITIATOR is connected to a TARGET the TARGET must maintain an active $\overline{BSY}$ signal. During DMA operations the $\overline{BSY}$ signal is monitored by the ASI and will halt operations if it goes inactive. To enable $\overline{BSY}$ to be monitored at other times the MR2 BSY bit (2) should be set. An interrupt will be generated if $\overline{BSY}$ goes inactive while MR2 BSY is set.

This interrupt sets bit 2 in the BSR.

4.7 COMMAND/MESSAGE/STATUS TRANSFERS

Command message and status bytes are transferred using programmed-I/O. The SCSI $\overline{REQ}$/$\overline{ACK}$ handshake is accomplished by monitoring and setting lines individually. Data is output via the ODR and read in via the CSD register.

The following code shows INITIATOR and TARGET programming for two of these cases. See Appendix A.1 for flowcharts.

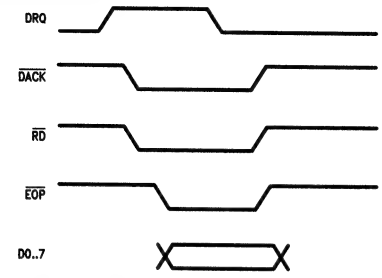
```
Initiator Command Send
{
   MR2 = monitor BSY
   TCR = Command Phase  /*02h*/
   while (bytes) to do) {
      while (REQ) inactive)
        idle; /*CSB bit 5 = 0*/
      if (BSR: phase match == 0)
        phase error;
      else {
        ODR = date byte;
        ICR = Assert ACK;
        while (REQ active)
          idle; /*CSB bit 5 == 1*/
        ICR = deassert ACK
        /* byte transfer complete */
        byte count --;
      }
   }
   goto data phase;
}

Target Message Receive
{
   /* assumed Assert BSY already set in ICR */
   MR2 = TARG MODE OR PARITY CHECK OR
         PARITY INTERRUPT;
   TCR = Message Out phase; /*06h*/
   delay (Bus Settle);
   TCR = Assert REQ;
   while (ACK inactive)
         idle; /* BSR bit 0 */
   data = CSD; /* parity is latched */
   if (BSR: parity error)
         error routine;
   else  {
         TCR = deassert REQ;
         while (ACK active)
            idle;
   }
    /* message done, can change to next
phase */
}
```

4.8 NON-BLOCK DMA TRANSFERS

Data transfers may be effected by DMA. This method should be used for optimum performance. Two methods of DMA are available-block and non-block mode. This section describes non-block mode transfers.

4.0 Device Operation (Continued)

The interface to the DMA controller uses the DRQ, $\overline{DACK}$, EOP lines in non-block mode. Each byte is requested ($\overline{DRQ}$) and ack'd ($\overline{DACK}$). Representative timing for a DMA read is shown in *Figure 4.8.1.*



TL/F/9756–7

FIGURE 4.8.1. Non-Block DMA Timing

4.8.1. NON-BLOCK DMA

DMA operation involves programming the ASI with the set-up parameters, initiating the DMA cycles and checking for correct operation when the completion interrupt is received. The DMA controller should be programmed with the data byte count and the memory start address. Methods of halting a DMA operation are covered in Section 4.11.

Setting up the ASI requires enabling or disabling the following: Data bus driving, DMA mode enable, $\overline{BSY}$ monitoring, $\overline{EOP}$ interrupt, parity checking, parity interrupt, TARGET Mode, bus phase.

Once set up DMA should be initiated by writing to address 5, 6, or 7 as appropriate. The DMA controller should assert $\overline{EOP}$ during the transfer of the last byte, although this may be done by the μP if the DMA transfers (n − 1) bytes and the μP transfers the last byte. See the application guide for more details (Section 7.0).

Upon completion the μP should check the following as required: End of DMA, Parity Error, Phase Match, Busy Error. The end of DMA occurs as a response to $\overline{EOP}$. SCSI transfers may still be underway so $\overline{REQ}$ and $\overline{ACK}$ must still be checked to establish when the final byte is finished.

The code below shows programming of the ASI in each of the four DMA cases. One of these cases is shown in a flow diagram in Appendix A.

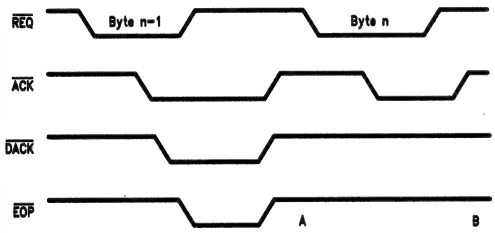
```
Initiator Send              /*DATA OUT PHASE*/
{
    Program DMA Controller;
    TCR = 00h;              /*phase*/
    ICR = 01h;              /*Assert_DBUS*/
    MR2 = 0Eh;
    SDS = 00;               /*Start DMA Send*/
    while (NOT interrupt)
      idle;
    while (CSD:REQ)
      idle                  /*wait for last
                              SCSI byte
                            transfer so phase
                              is checked*/
    if(BSR:Busy error OR NOT
        (BSR:End_of_DMA))
      error routine;
    else {                  /*DMA End*/
      MR2 = 04h;            /*reset DMA bit*/
      ICR = 0;
    }
}

Initiator Receive           /*DATA IN PHASE*/
{
    Program DMA Controller;
    TCR = 01h;              /*phase*/
    MR2 = 3Eh;
    SDI = 0;                /*Start DMA Init
                              Rx*/
    while (NOT interrupt)
        idle;
/*no need to wait for last SCSI handshake
  done since DMA done implies it is
  checked*/
    if(BSR:parity_error OR BSR: busy_error
     or NOT (BSR End of DMA)
       do error routines;
    else {                  /*End of DMA*/
      while (CSD:REQ)
       idle; /*wait for REQ inactive
          to deassert ACK*/
      MR2 = 04h;
    }
}

Target Receive              /*DATA OUT PHASE*/
{
    Program DMA Controller;
    TCR = 0;                /*phase*/
    ICR = 08h;
    MR2 = 7Ah;              /*check parity*/
    SDT = 0;                /*Start DMA Targ Rx*/
    while (not interrupt)
        idle;
    /*when End of DMA occurs the last byte
     has been read and checked*/
    if(BSR:parity_error OR NOT(BSR: End_of_DMA)
      error routine;
        {       else        /*End of DMA*/
      while (BSR:ACK)
            idle;
    /*Not True End of DMA, so wait until SCSI
     bus inactive before changing phase*/
    MR2 = 40h;
    change phase as required;
    }
}
```

4.0 Device Operation (Continued)

```
Target Send              /*DATA IN PHASE*/
{
    Program DMA Controller;
    TCR = 01h;           /*phase*/
    ICR = 09h;
    MR2 = 4Ah;
    SDS = 0;             /*Start DMA Send*/
    while (NOT interrupt)
        idle;
    if(NOT(BSR:End_of_DMA)
        error;
    else {   /*DMA end*/
      repeat {
       while (CSB:REQ OR BSR:ACK)
       loop count = 3;
       loop count --;} /*decrement*/
      until (loop count == 0);
      MR2 = 40h;
      Change phase as required;
      }
}
```

Some explanation of the final part of Target Send is required. In this type of DMA operation it is very difficult to exactly determine the True End of DMA simply detecting $\overline{REQ}$ and $\overline{ACK}$ simultaneously inactive is not enough.

Reference to *Figure 4.8.2* will help to understand the following text.



TL/F/9756–8

FIGURE 4.8.2. Target Send DMA

As shown in *Figure 4.8.2* $\overline{ACK}$ going active causes the DRQ for the next byte and also $\overline{REQ}$ to go inactive. $\overline{ACK}$ going inactive allows $\overline{REQ}$ to go active for the next byte. If the INITIATOR is slow removing $\overline{ACK}$ the μP may sample the SCSI bus after the $\overline{EOP}$ interrupt at point A. Here both $\overline{REQ}$ and $\overline{ACK}$ will be inactive, but there is one more byte to transfer on SCSI. Due to chip timing delays this condition will not last more than 200 ns. A safe way to determine the True End of DMA is to sample $\overline{REQ}$ and $\overline{ACK}$ and ONLY when both are inactive in three successive samples will the μP be at point B in the figure.

4.9 BLOCK MODE DMA TRANSFERS

In Block Mode the DMA interface uses the DRQ, $\overline{DACK}$, $\overline{EOP}$ and READY lines, DRQ is asserted once at the beginning of transfers and deasserted once $\overline{DACK}$ is received. $\overline{DACK}$ should be asserted continuously for the duration of all the transfer. $\overline{EOP}$ should be asserted during the last DMA byte signal when the next DMA byte transfers. The ASI asserts the READY signal when the next DMA byte should be transferred.

As for non-block mode the End of DMA interrupt is just $\overline{EOP}$, also in block mode receive the ASI does not return READY to an active signal after $\overline{EOP}$. This means external logic must gate off READY if the μP is not to be locked up. For more details see Section 7.0.

The block mode is intended for systems where the overhead of handing the system busses to and from the μP and DMA controller is too great. The block mode handshake is not necessarily faster than non-block (it may be) but the overall transfer rate is improved once the bus exchange overhead is removed. Of course the μP is prevented from executing for the whole DMA operation.

If a phase mismatch occurs the READY signal is left in the inactive state. The DMA controller must hand back the bus to the μP and the inactive READY signal may need to be gated off.

When performing DMA as an INITIATOR the $\overline{EOP}$ signal does not deassert $\overline{ACK}$ on the SCSI bus. Firmware must determine when $\overline{REQ}$ is inactive after the last SCSI transfer then reset the MR2 DMA bit to deassert $\overline{ACK}$.

Programming the ASI in block mode is the same as non-block mode except bit 7 in MR2 should also be set.

4.10 PSEUDO DMA

The system design can utilize ASI DMA logic for non-data transfers. This removes the need to poll $\overline{REQ}$/$\overline{ACK}$ and program the assertion/deassertion of the handshake signal. The μP can emulate a DMA controller by asserting $\overline{DACK}$ and $\overline{EOP}$ signals. DRQ may be sampled by reading the BSR. In most cases the chip decode logic can be adapted to this use for little or no cost. See Section 7.0 for further details.

4.11 HALTING A DMA OPERATION

There are three ways to halt a DMA operation apart from a chip or SCSI reset. These methods are: $\overline{EOP}$, phase mismatch and resetting the DMA MODE bit in MR2.

4.11.1 End Of Process

$\overline{EOP}$ is asserted for a minimum period during the last DMA cycle. The $\overline{EOP}$ signal generates the End of DMA interrupt. $\overline{EOP}$ does not cause the MR2 DMA mode bit to be reset.

4.11.2 DMA Phase Mismatch

If a $\overline{REQ}$ goes active while there is a phase mismatch the DMA will be halted and an interrupt generated. The ASI will stop driving the SCSI bus when the mismatch occurs. A phase mismatch is when the TCR phase bits do not match the SCSI bus values.

4.11.3 DMA Mode Bit

If $\overline{EOP}$ is not used the best method is to reset the MR2 DMA Mode bit. This bit may be reset at any time, and should be reset after an End of DMA interrupt or a phase mismatch.

4.0 Device Operation (Continued)

Resetting the bit disables all DMA logic and thus should only be reset at the True End of DMA condition. Additionally all DMA logic is reset so this bit must be reset then set again to carry out the next DMA phase.

5.0 Interrupts

5.1 OVERVIEW

Before individually describing each interrupt an explanation of the use of interrupts is required.

5.2 USING INTERRUPTS

Interrupts are controlled by bits in MR2 if control is provided. Not all interrupts can be disabled under software control. When an interrupt occurs both the BSR and CSD register must be read and analysed to determine the source of interrupt. Since status is NOT provided for each interrupt great care should be exercised when determining the interrupt source.

5.3 SCSI PARITY ERROR

If SCSI parity checking is enabled via MR2 bit 5 an interrupt can occur as a result of a read from CSD, a selection/(re)selection, or a DMA receive operation. The parity error bit (bit 5) in the BSR will be set if checking is enabled. An interrupt will occur if Enable Parity Interrupt (bit 4) of MR2 is set. The interrupt is reset by reading HA7. Following an interrupt the BSR and CSD should contain the values shown below.

Bit 7							Bit 0
x	x	1	1	x	x	x	x
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	1	x	x	x	x	0	x
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSD

5.4 END OF DMA

If $\overline{EOP}$ is asserted during a DMA transfer bit 7 of the BSR will be set and an interrupt generated if bit 3 of MR2 is 1. $\overline{EOP}$ is recognized when $\overline{EOP}$, $\overline{DACK}$ and either $\overline{IOR}$ or $\overline{IOW}$ are all simultaneously active for a minimum period. The interrupt may be reset by reading HA 7. Following an interrupt the BSR and CSD should contain the values shown below.

Bit 7							Bit 0
1	x	x	1	x	x	0	x
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	1	x	x	x	x	0	x
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSD

5.5 DMA PHASE MISMATCH

When the SCSI $\overline{REQ}$ goes active during a DMA operation the contents of the TCR are compared with the SCSI phase lines C/$\overline{D}$, $\overline{MSG}$ and I/$\overline{O}$. If the two do not match an interrupt is generated. This interrupt will occur as long as the MR2 DMA bit is set (bit 1), i.e. it cannot be masked. The mismatch removes the ASI from driving the SCSI data bus. The interrupt may reset by reading HA 7. Following an interrupt the BSR and CSD should contain the values shown below.

Bit 7							Bit 0
x	0	x	1	0	x	x	x
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	x	x	x	x	x	0	x
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSD

5.6 BUSY LOSS

If bit 2 MR2 is set the SCSI $\overline{BSY}$ signal is monitored and an interrupt is generated if $\overline{BSY}$ is continuously inactive for at least a BUS SETTLE DELAY (400 ns). This interrupt may be reset by reading HA 7. Following an interrupt the BSR and CSD should contain the values shown below, where usually CSD = 00.

Bit 7							Bit 0
x	x	x	1	x	1	0	x
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	0	x	x	x	x	x	x
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSD

5.7 (RE)SELECTION

An interrupt will be generated when: $\overline{SEL}$ is active, $\overline{BSY}$ is inactive, and the device ID is true. The device ID is determined by the value in the SER. If ANY non-zero bit in the SER has its corresponding SCSI data bit active during selection the device ID is true. If I/$\overline{O}$ is active this is a reselection. The interrupt is disabled by writing all zeros to the SER, and reset by reading HA 7.

5.0 Interrupts (Continued)

If SCSI parity checking is enabled it will be checked and should be valid. Following an interrupt the BSR and CSD should contain the values shown below.

Bit 7							Bit 0
0	0	0	1	x	0	x	0
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

BSR

Bit 7							Bit 0
0	0	0	0	0	0	1	x
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	C/$\overline{D}$	I/$\overline{O}$	$\overline{SEL}$	$\overline{DBP}$

CSD

6.0 Reset Conditions

6.1 GENERAL

There are three ways to reset the ASI; µP chip $\overline{RESET}$, SCSI bus reset applied externally, SCSI bus reset issued by the ASI.

6.2 CHIP RESET

When the $\overline{RESET}$ signal is asserted for the required duration the ASI clears ALL internal registers and therefore resets all logic. This action does not create an interrupt or generate a SCSI reset.

6.3 EXTERNAL SCSI RESET

When a SCSI $\overline{RST}$ is applied externally the ASI resets all registers and logic and issues an interrupt. The only register bits not affected are the Assert RST bit (bit 7) in the ICR and the TARGET Mode bit (bit 6) in MR2.

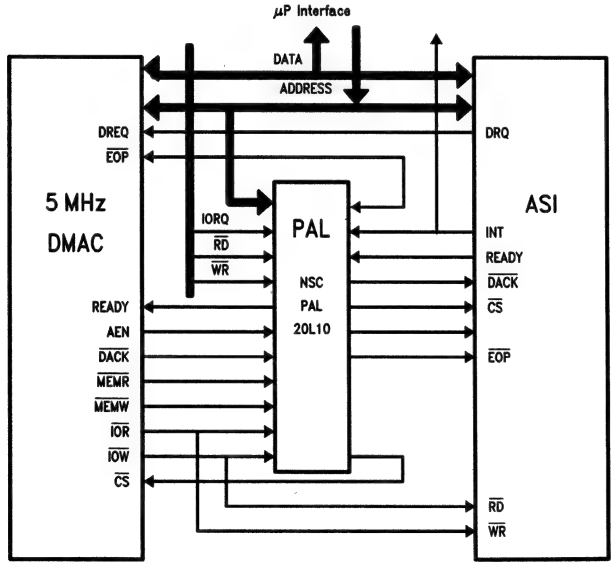
6.4 SCSI RESET ISSUED

When the µP sets the Assert RST bit in the ICR the $\overline{RST}$ signal goes active. Since the ASI monitors $\overline{RST}$ also the same reset actions as in 6.3 apply. The SCSI $\overline{RST}$ signal will remain active as long as bit 7 in the ICR is set—i.e. until programmed 0 or a chip $\overline{RESET}$ occurs.

7.0 Application Guide

This section is intended to show the interface between the µP, ASI and DMA controller (DMAC). *Figure 7.1* shows a general interface when the ASI and DMAC are I/O-mapped devices. This configuration will implement a 2 to 2.5M Bytes/sec SCSI port using 2 cycle compressed timing from the 5 MHz DMAC.

Using a faster DMAC and memory may allow the ASI to operate at a higher rate—but of course any system will be limited by the available DMA rate from the SCSI device currently connected to. The interface shown has several features that are examined more closely in the following text.



TL/F/9756–9

FIGURE 7.1. µP/ASI/DMA Interface

7.0 Application Guide (Continued)

All the interface signal requirements are satisfied by a PAL device. The memory interface is not shown, only the relevant DMAC and μP lines are included.

The ASI data and address lines connect directly to the μP/DMAC busses. The DRQ output from the ASI goes direct to the DMAC. The $\overline{EOP}$ output from the DMAC goes to the ASI input, but can also be asserted via the PAL since the DMAC output is open-drain.

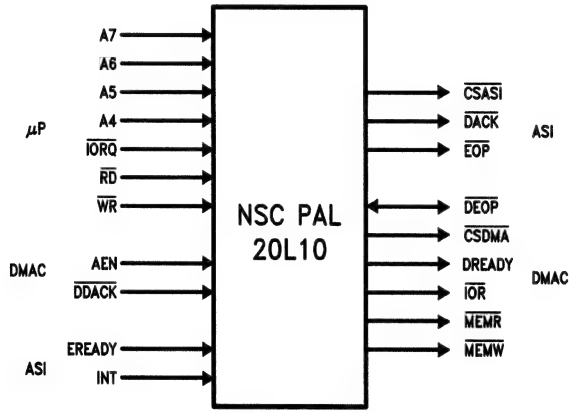
The PAL is programmed so that the μP can access the ASI in three ways. The three access types are: Register R/W, DMA R/W, DMA with $\overline{EOP}$. Examination of the PAL equations below shows how the μP may perform any of the three basic access types simply by accessing the ASI at different I/O address slots. This enables the μP to simulate a DMAC (pseudo-DMA). DMA mode may then be used for all information transfer phases.

In DMA mode the ASI generates all SCSI handshakes. At all other times the μP is responsible for $\overline{REQ}/\overline{ACK}$ handshakes. Using pseudo-DMA may reduce μP overhead.

When doing DMA transfers via BLOCK MODE and an error occurs, the ASI may not deassert the READY signal. For some DMA controllers this may lock the bus, so the PAL asserts READY and $\overline{EOP}$ to the DMA if an interrupt occurs while READY is false. This completes the current DMA cycle and prevents further DMA for the rest of the block thus allowing the bus to be handed back to the μP for servicing.

The PAL generates $\overline{RD}$ and $\overline{WR}$ strobes while the μP is bus master, but the DMAC provides the strobes while it is bus master so the PAL outputs are TRI-STATE.

The PAL details are shown in *Figure 7.2* with the signal definitions and equations following.



TL/F/9756–10

FIGURE 7.2. Interface PAL

7.0 Application Guide (Continued)

```
CSASI = IORQ*A7*A6*A5*A4*AEN; ASI reg R/W chip select
ADACK = IORQ*A7*A6*A5*A4*RD;  µP pseudo-DMA cycle
        IORQ*A7*A6*A5*A4*WR
        +IORQ*A7*A6*A5*A4*RD; µP pseudo-DMA with EOP
        +IORQ*A7*A6*A5*A4*WR
        +DDACK;               ; DMAC DMA cycle
IF(AEN)AEOP = IORQ*A7*A6*A5*A4*RD; µP pseudo-DMA with EOP
            +IORQ*A7*A6*A5*A4*WR + DEOP* AREADY
IF(DDACK*AREADY*INT)DEOP = DDACK*AREADY*INT
                          ;DMA cycle with error
CSDMA = IORQ*A7*A6*A5*A4    ;DMAC register R/W
DREADY = AREADY*INT         ;ASI not READY and not INT
       +AREADY*DDACK        ;ASI not READY and DMA cycle active
IF(AEN)IOR  = IORQ*RD       ;µP I/O Read cycle
IF(AEN)IOW  = IORQ*WR       ;µP I/O Write cycle
IF(AEN)MEMR = IORQ*RD       ;µP memory Read cycle
IF(AEN)MEMW = IORQ*WR       ;µP memory Write cycle
```

FIGURE 7.3. PAL Equations

The µP and DMA signals are defined below

A7–A4	Address bus
IORQ	Memory I/O cycle select
RD	Read Strobe
WR	Write Strobe
AEN	High DMA address enable asserted by DMAC
DDACK	DMAC DMA Acknowledge
CSDMA	DMA Chip Select
DREADY	Ready signal to DMAC—inserts wait-states when low
IOR, IOW	I/O data strobes to/from DMAC
MEMR, MEMW	Memory data strobe from DMAC

8.0 Absolute Maximum Ratings*

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7.0V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} + 0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} + 0.5V
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D)	500 mW
Lead Temperature (T_L) (Soldering, 10 sec)	260°C
Electro-Static Discharge Rating	2 kV

*Absolute maximum ratings are those values beyond which damage to the device may occur.

9.0 DC Electrical Characteristics (V_{CC} = 5.0V ± 10% unless otherwise specified) T_A = 0°C to +70°C

Symbol	Parameter	Conditions	Typ	Limit	Units
V_{IH}	Minimum High Level Input Voltage			2.0	V
V_{IL}	Maximum Low Level Input Voltage			0.8	V
V_{OH1} V_{OH2}	Minimum High Level Output Voltage	$\|I_{OUT}\|$ = 20 μA $\|I_{OUT}\|$ = 4.0 mA		V_{CC} − 0.1 2.4	V V
V_{OL1} V_{OL2} V_{OL3}	Maximum Low Level Output Voltage	SCSI Bus Pins: $\|I_{OL}\|$ = 48 mA Other Pins: $\|I_{OL}\|$ = 20 μA $\|I_{OL}\|$ = 8.0 mA		0.5 0.1 0.4	V V V
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND		±1	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current	V_{OUT} = V_{CC} or GND		±10	μA
I_{CC}	Supply Current	V_{IN} = V_{CC} or GND SCSI Inputs = 3V	2.5	4	mA

Capacitance T_A = 25°C, f = 1 MHz

Symbol	Parameter (Note 3)	Typ	Units
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	7	pF

AC Test Conditions

Input Pulse Level	GND to 3.0V
Input Rise and Fall Times	6 ns
Input/Output Reference Levels	1.3V
TRI-STATE Reference Levels (Note 2)	Active Low + 0.5V Active High − 0.5V

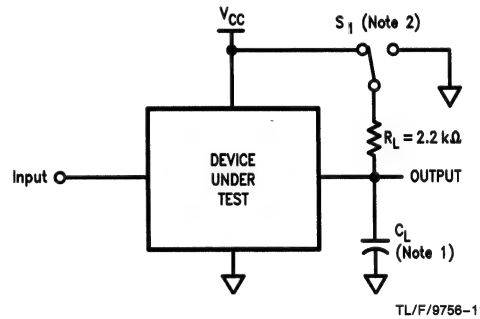
Note 1: C_L = 50 pF including jig and scope capacitance.

Note 2: S1 = Open for push-pull outputs.

S1 = V_{CC} for active low to TRI-STATE.

S1 = GND for active high to TRI-STATE.

Note 3: This parameter is not 100% tested.

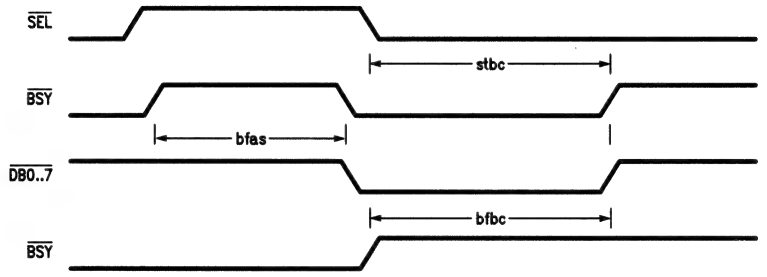


TL/F/9756–11

10.0 AC Electrical Characteristics all parameters are preliminary and subject to change without notice

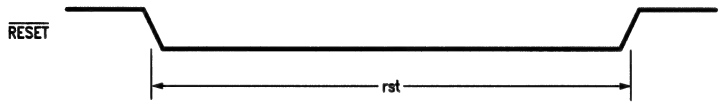
Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
bfas	BSY False to Arbitrate Start	1200		2200	ns
bfbc	BSY False to Bus Clear			800	ns
rst	RESET Pulse Width	150			ns
stbc	SEL True to Bus Clear			500	ns

10.1 ARBITRATION



TL/F/9756–12

10.2 μP RESET



TL/F/9756–13

10.0 AC Electrical Characteristics

all parameters are preliminary and subject to change without notice (Continued)

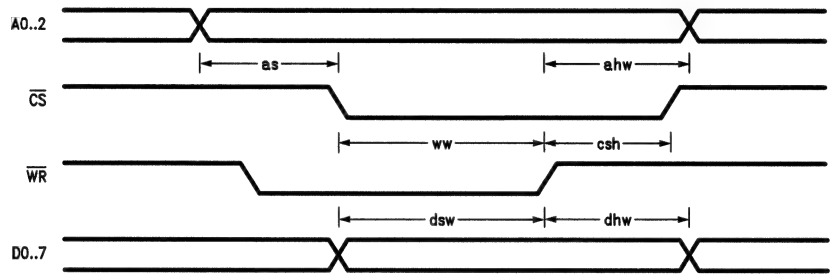
Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
ahr	Address Hold from End of Read Enable (Note 1)	10			ns
ahw	Address Hold from End of Write Enable (Note 2)	10			ns
as	Address Setup to Read or Write Enable (Notes 1, 2)	10			ns
csh	CS Hold from End of RD or WR	0			ns
dhr	Data Hold from End of Read Enable (Notes 1, 3)	10		60	ns
dhw	μP Data Hold Time from End of WR	20			ns
dsw	Data Setup to End of Write Enable	50			ns
rdv	Data Valid from Read Enable (Note 1)			100	ns
ww	Write Enable Width (Note 2)	60			ns

Note 1: Read enable (μP) is CS and RD active.

Note 2: Write enable (μP) is CS and WR active.

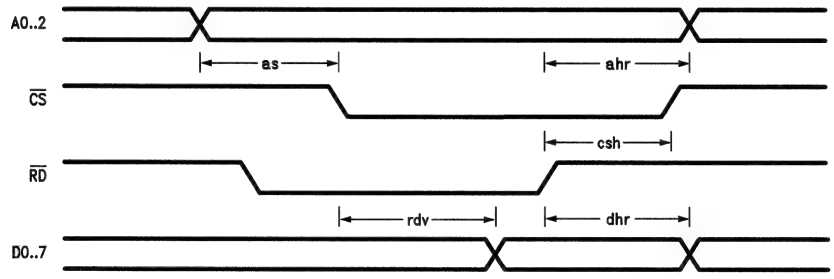
Note 3: This includes the RC delay inherent in the tests' method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

10.3 μP WRITE



TL/F/9756–14

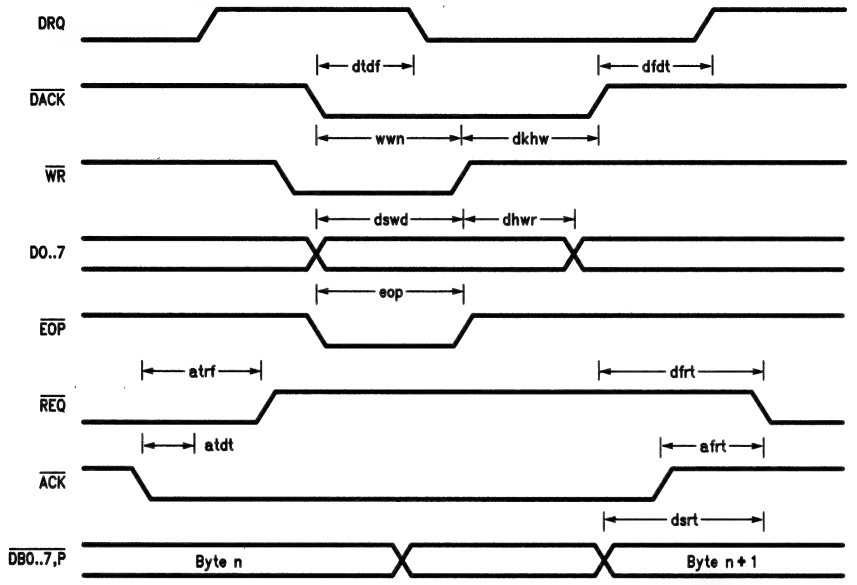
10.4 μP READ



TL/F/9756–15

10.0 AC Electrical Characteristics (Continued)

10.5 DMA WRITE (NON-BLOCK MODE) TARGET SEND



TL/F/9756–16

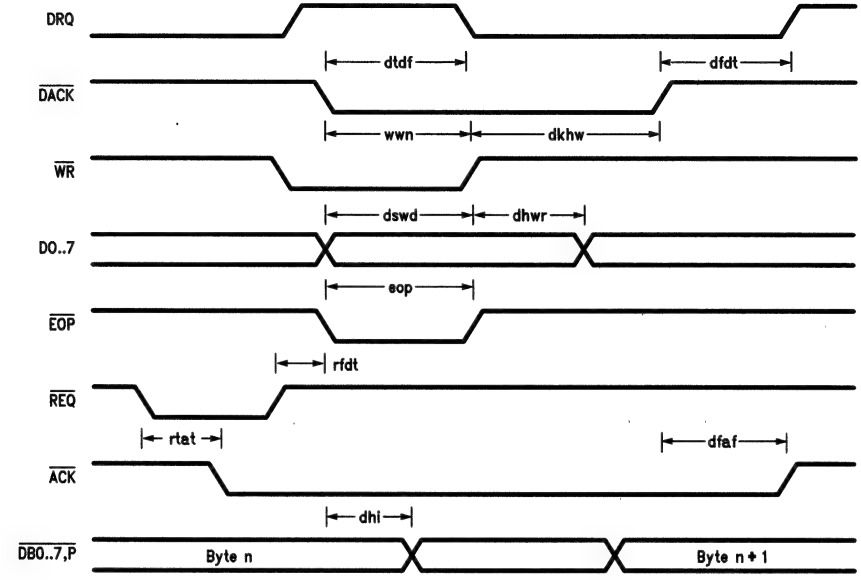
Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
afrt	ACK False to REQ True (DACK or WR False)			120	ns
atdt	ACK True to DRQ True (Target)			90	ns
atrf	ACK True to REQ False (Target)			115	ns
dfdt	DACK False to DRQ True	30	90		ns
dfrt	DACK False to REQ True (ACK False)			110	ns
dhwr	DMA Data Hold Time from End of WR	30			ns
dkhw	DACK Hold from End of WR	0			ns
dsrt	SCSI Data Setup to REQ True (Target Send) (Note 1)	40			ns
dswd	Data Setup to End of DMA Write Enable	50			ns
dtdf	DACK True to DRQ False			100	ns
eop	Width of EOP Pulse (Note 2)	40			ns
wwn	DMA Non-Block Mode Write Enable Width (Note 3)	60			ns

Note 1: EOP, DACK, RD/WR must all be true for recognition of EOP.

Note 2: Write enable (DMA) is DACK and WR active.

10.0 AC Electrical Characteristics (Continued)

10.6 DMA WRITE (NON-BLOCK MODE) INITIATOR SEND



TL/F/9756–17

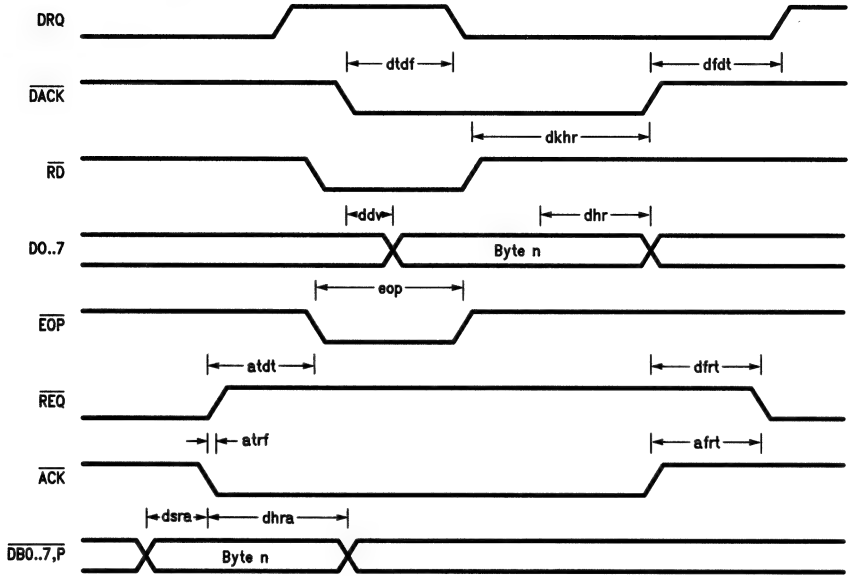
Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
dfaf	DACK False to ACK False (Non-Block Initiator Send)			120	ns
dfdt	DACK False to DRQ True	30	90		ns
dhi	SCSI Data Hold from Write Enable—Initiator	15			ns
dhwr	DMA Data Hold Time from End of WR	30			ns
dkhw	DACK Hold from End of WR	0			ns
dswd	Data Setup to End of DMA Write Enable	50			ns
dtdf	DACK True to DRQ False			100	ns
eop	Width of EOP Pulse (Note 1)	40			ns
rfdt	REQ False to DRQ True			100	ns
rtat	REQ True to ACK True (Initiator Send)			100	ns
wwn	DMA Non-Block Mode Write Enable Width (Note 2)	60			ns

Note 1: EOP, DACK, RD/WR must all be true for recognition of EOP.

Note 2: Write enable (DMA) is DACK and WR active.

10.0 AC Electrical Characteristics (Continued)

10.7 DMA READ (NON-BLOCK MODE) TARGET RECEIVE



TL/F/9756–18

Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
afrt	ACK False to REQ True (DACK or WR False)			120	ns
atdt	ACK True to DRQ True (Target)			90	ns
atrf	ACK True to REQ False (Target)			115	ns
ddv	DMA Data Valid from Read Enable (Note 1)			90	ns
dfdt	DACK False to DRQ True	30	90		ns
dfrt	DACK False to REQ True (ACK False)			110	ns
dhr	Data Hold from End of Read Enable (Notes 1, 2)	10		60	ns
dhra	SCSI Data Hold from REQ or ACK True (Receive)	30			ns
dkhr	DACK Hold from End of RD	0			ns
dsra	SCSI Data Setup Time to REQ or ACK True (Receive)	20			ns
dtdf	DACK True to DRQ False			100	ns
eop	Width of EOP Pulse (Note 4)	40			ns

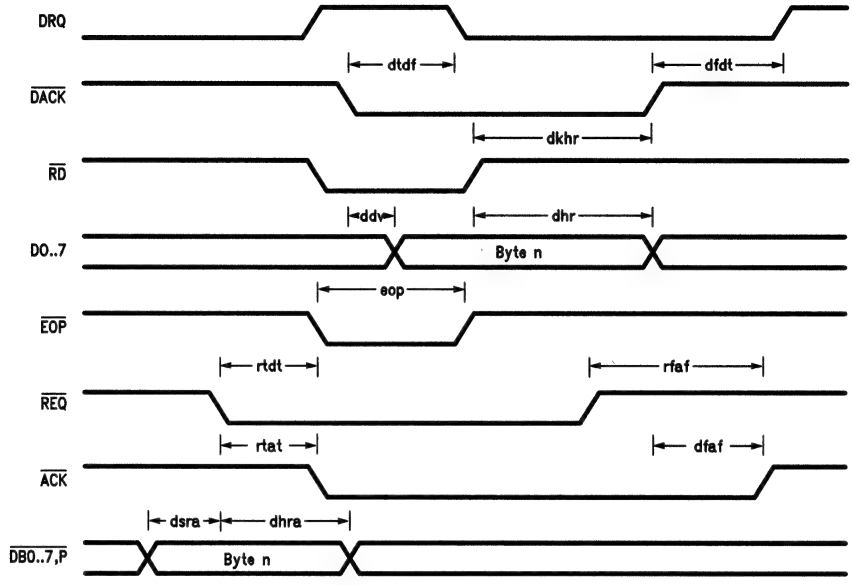
Note 1: Read enable (DMA) is DACK and RD active.

Note 2: This includes the RC delay inherent in the tests' method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

Note 3: EOP, DACK, RD/WR must all be true for recognition of EOP.

10.0 AC Electrical Characteristics (Continued)

10.8 DMA READ (NON-BLOCK MODE) INITIATOR RECEIVE



TL/F/9756–19

Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
ddv	DMA Data Valid from Read Enable (Note 1)			90	ns
dfaf	DACK False to ACK False (REQ False, Non-block, In rx)			120	ns
dfdt	DACK False to DRQ True	30	90		ns
dhr	Data Hold from End of Read Enable (Notes 1, 2)	10		60	ns
dhra	SCSI Data Hold from REQ or ACK True (Receive)	30			ns
dkhr	DACK Hold from End of RD	0			ns
dsra	SCSI Data Setup Time to REQ or ACK True (Receive)	20			ns
dtdf	DACK True to DRQ False			100	ns
eop	Width of EOP Pulse (Note 3)	40			ns
rfaf	REQ False to ACK False (DACK False)			100	ns
rtat	REQ True to ACK True (Initiator Receive)			100	ns
rtdt	REQ True to DRQ True			120	ns

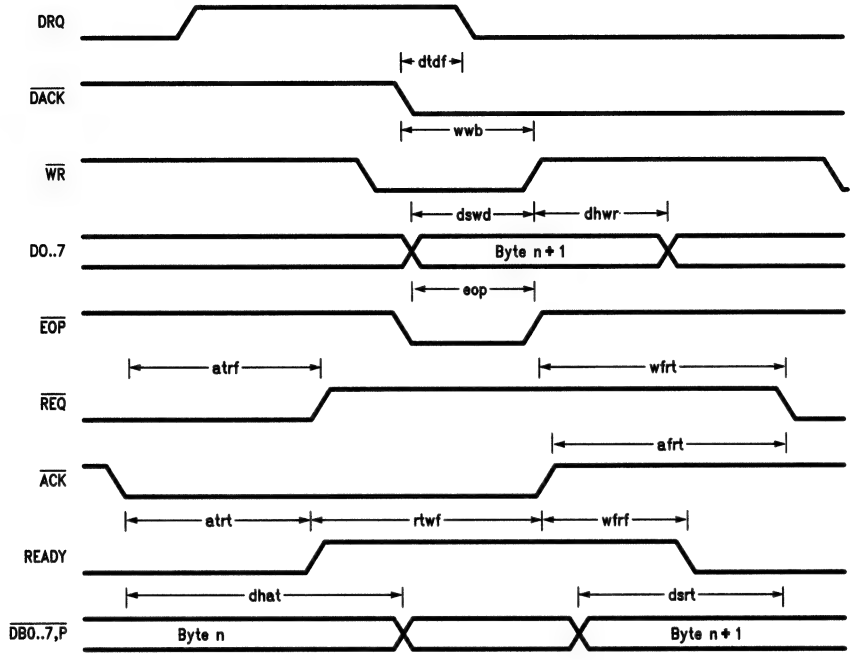
Note 1: Read enable (DMA) is DACK and RD active.

Note 2: This includes the RC delay inherent in the tests' method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

Note 3: EOP, DACK, RD/WR must all be true for recognition of EOP.

10.0 AC Electrical Characteristics (Continued)

10.9 DMA WRITE (BLOCK MODE) TARGET SEND



TL/F/9756–20

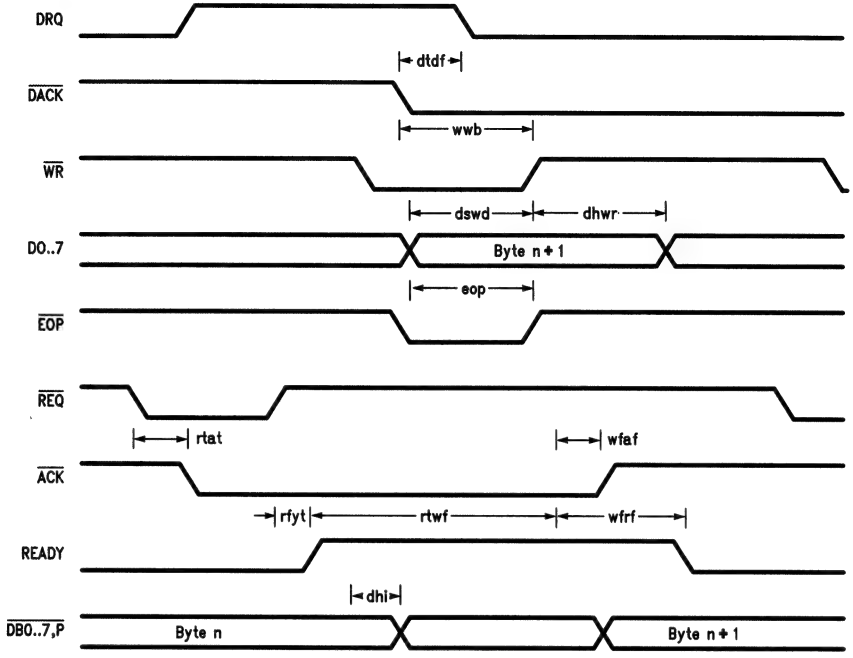
Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
afrt	ACK False to REQ True (DACK or WR False)			120	ns
atrf	ACK True to REQ False (Target)			115	ns
atrt	ACK True to READY True (Block Mode Target Send)			110	ns
dhat	SCSI Data Hold from ACK True	40			ns
dhwr	DMA Data Hold Time from End of WR	30			ns
dsrt	SCSI Data Setup to REQ True	50			ns
dswd	Data Setup to End of DMA Write Enable	50			ns
dtdf	DACK True to DRQ False			100	ns
eop	Width of EOP Pulse (Note 1)	40			ns
rtwf	READY true to WR False	60			ns
wfrf	WR False to READY False			100	ns
wfrt	WR False to REQ True (ACK False)			120	ns
wwb	DMA Write Enable Width (Note 2)	60			ns

Note 1: EOP, DACK, RD, WR must all be true for recognition of EOP.

Note 2: Write enable (DMA) is DACK and WR active.

10.0 AC Electrical Characteristics (Continued)

10.10 DMA WRITE (BLOCK MODE) INITIATOR SEND



TL/F/9756–21

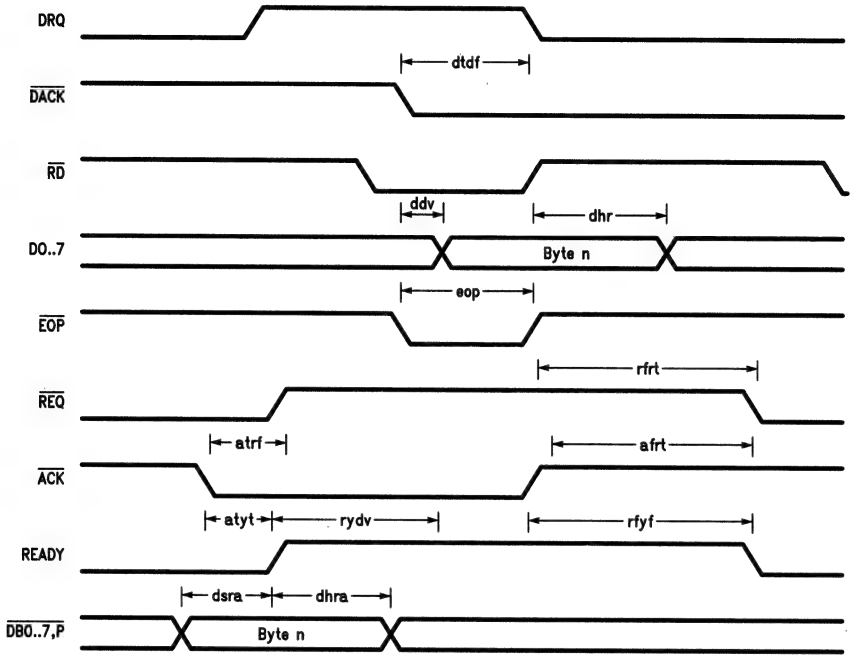
Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
dhi	SCSI Data Hold from Write Enable—Initiator	15			ns
dhwr	DMA Data Hold Time from End of WR	30			ns
dswd	Data Setup to End of DMA Write Enable	50			ns
dtdf	DACK True to DRQ False			100	ns
eop	Width of EOP Pulse (Note 2)	40			ns
rfyt	REQ False to READY True			80	ns
rtat	REQ True to ACK True			100	ns
rtwf	READY True to WR False	60			ns
wfaf	WR False to ACK False (REQ False)			120	ns
wfrf	WR False to READY False			100	ns
wwb	DMA Write Enable Width (Note 1)	60			ns

Note 1: Write enable (DMA) is DACK and WR active.

Note 2: EOP, DACK, RD/WR must all be true for recognition of EOP.

10.0 AC Electrical Characteristics (Continued)

10.11 DMA READ (BLOCK MODE) TARGET RECEIVE



TL/F/9756–22

Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
afrt	ACK False to REQ True (DACK or WR False)			120	ns
atrf	ACK True to REQ False			115	ns
atyt	ACK True to READY True			110	ns
ddv	DMA Data Valid from Read Enable (Note 1)			90	ns
dhr	Data Hold from End of Read Enable (Notes 1, 2)	10		60	ns
dhra	SCSI Data Hold from REQ or ACK True	30			ns
dsra	SCSI Data Setup Time to REQ or ACK True	20			ns
dtdf	DACK True to DRQ False			100	ns
eop	Width of EOP Pulse (Note 3)	40			ns
rfrt	RD False to REQ True (ACK False)			100	ns
rfyf	RD False to READY False			110	ns
rydv	READY True to Data Valid			35	ns

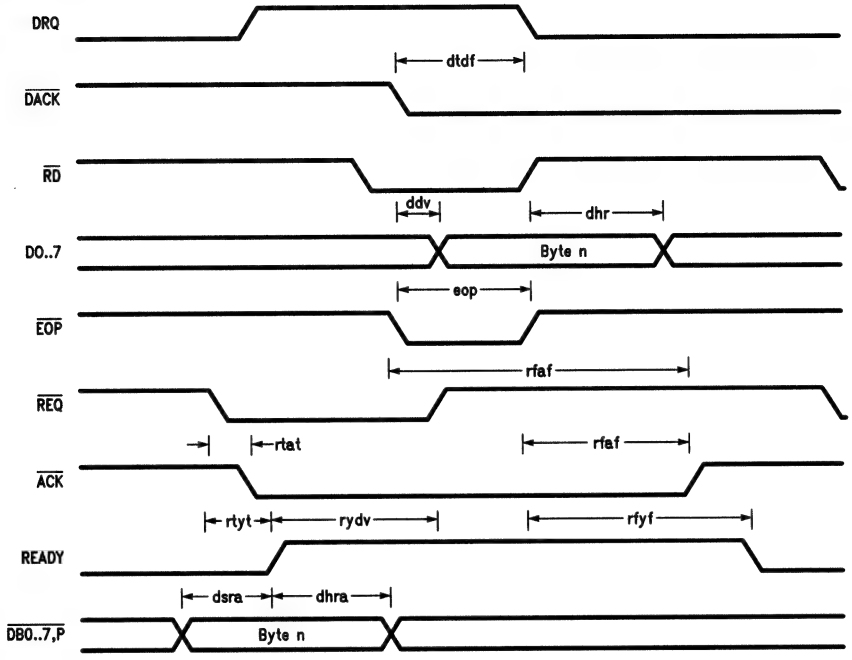
Note 1: Read enable (DMA) is DACK and RD active.

Note 2: This includes the RC delay inherent in the tests' method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

Note 3: EOP, DACK, RD/WR must all be active for recognition of EOP.

10.0 AC Electrical Characteristics (Continued)

10.12 DMA READ (BLOCK MODE) INITIATOR RECEIVE



TL/F/9756–23

Symbol	Parameter	DP5380			Units
		Min	Typ	Max	
ddv	DMA Data Valid from Read Enable (Note 1)			90	ns
dhr	Data Hold from End of Read Enable (Notes 1, 2)	10		60	ns
dhra	SCSI Data Hold from REQ or ACK True	30			ns
dsra	SCSI Data Setup Time to REQ or ACK True	20			ns
dtdf	DACK True to DRQ False			100	ns
eop	Width of EOP Pulse (Note 3)	40			ns
rdaf	RD False to ACK False (REQ False)			125	ns
rfaf	REQ False to ACK False (DACK False)			100	ns
rfyf	RD False to READY True False			110	ns
rtat2	REQ True to ACK True			100	ns
rtyt	REQ True to READY True			75	ns
rydv	READY True to Data Valid			35	ns

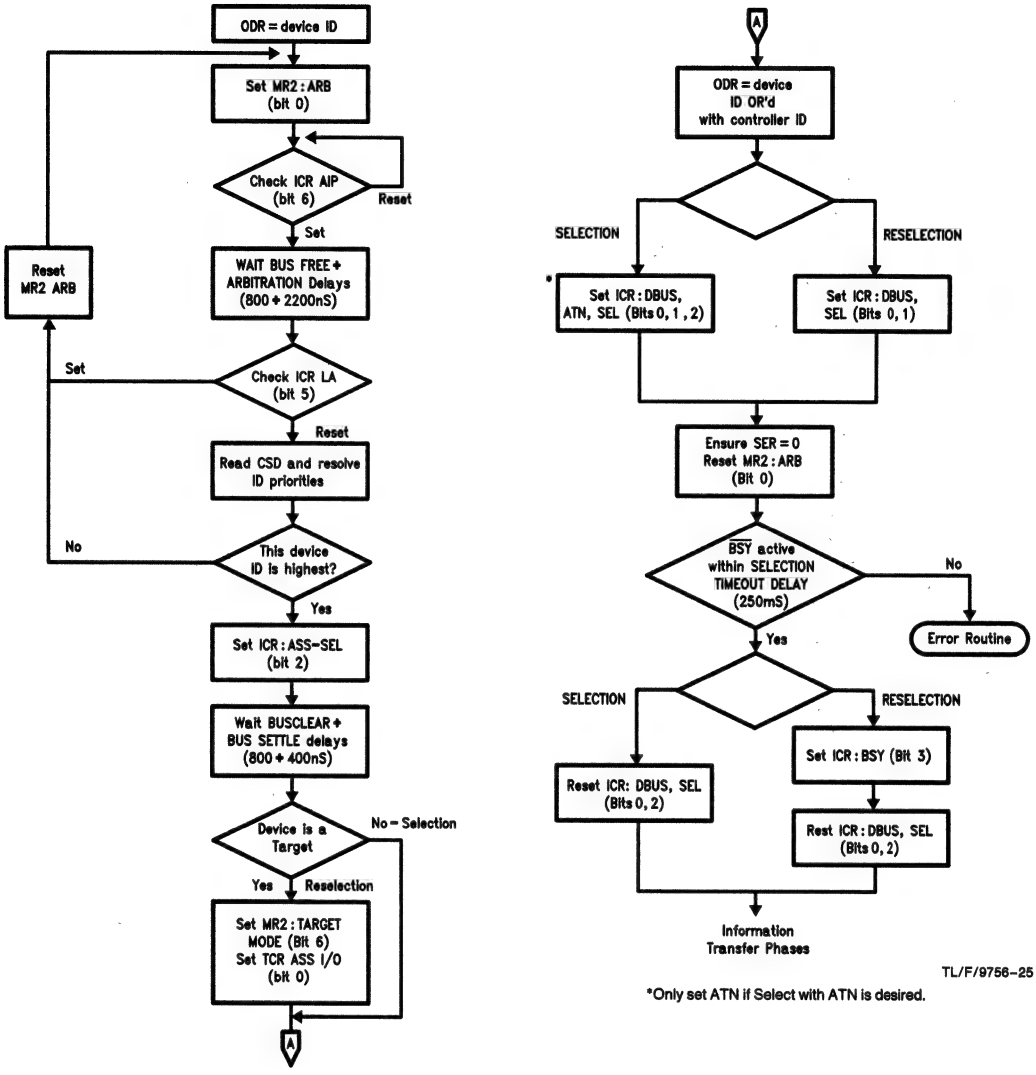
Note 1: Read enable (DMA) is DACK and RD active.

Note 2: This includes the RC delay inherent in the tests' method. These signals typically turn off after 25 ns enabling other devices to drive these lines with no contention.

Note 3: EOP, DACK, RD/WR must all be active for recognition of EOP.

Appendix A1

Arbitration and (Re)Selection



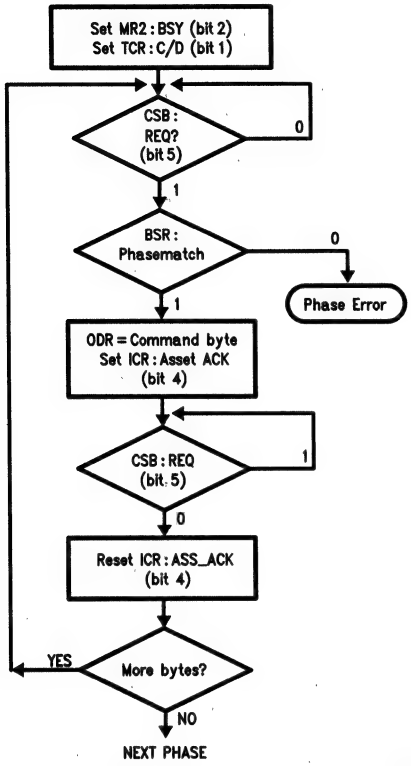
TL/F/9756–24

TL/F/9756–25

*Only set ATN if Select with ATN is desired.

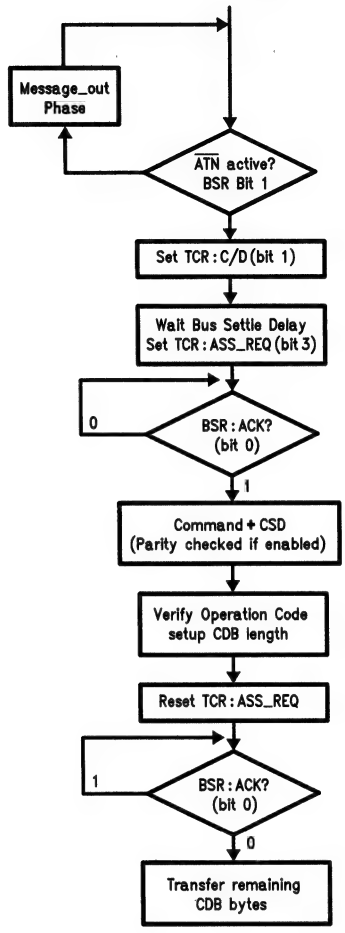
Appendix A1 (Continued)

Command Transfer (Initiator)



TL/F/9756–26

Command Transfer (Target)



TL/F/9756–27

Appendix A2

Register Chart

READ

Current SCSI Data (CSD)

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Initiator Command Register (ICR)

Bit 7							Bit 0
$\overline{RST}$	AIP	LA	$\overline{ACK}$	$\overline{BSY}$	$\overline{SEL}$	$\overline{ATN}$	DBUS

Mode Register 2 (MR2)

Bit 7							Bit 0
BLK	TARG	PCHK	PINT	EOP	BSY	DMA	ARB

Target Command Register (TCR)

Bit 7							Bit 0
0	0	0	0	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$

Current SCSI Bus Status (CSB)

Bit 7							Bit 0
$\overline{RST}$	$\overline{BSY}$	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$	$\overline{SEL}$	$\overline{DBP}$

Bus and Status Register (BSR)

Bit 7							Bit 0
EDMA	DRQ	SPER	INT	PHSM	BSY	$\overline{ATN}$	$\overline{ACK}$

Input Data Register (IDR)

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Reset Parity/Interrupt (RPI)—Mode N

Bit 7							Bit 0
x	x	x	x	x	x	x	x

X = Unknown

WRITE

Output Data Register (ODR)

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Initiator Command Register (ICR)

Bit 7							Bit 0
$\overline{RST}$	TEST	DIFF EN	$\overline{ACK}$	$\overline{BSY}$	$\overline{SEL}$	$\overline{ATN}$	$\overline{DBUS}$

Mode Register 2 (MR2)

Bit 7							Bit 0
BLK	TARG	PCHK	PINT	EOP	BSY	DMA	ARB

Target Command Register (TCR)

Bit 7							Bit 0
x	x	x	x	$\overline{REQ}$	$\overline{MSG}$	$\overline{C/D}$	$\overline{I/O}$

Select Enable Register (SER)

Bit 7							Bit 0
$\overline{DB7}$	$\overline{DB6}$	$\overline{DB5}$	$\overline{DB4}$	$\overline{DB3}$	$\overline{DB2}$	$\overline{DB1}$	$\overline{DB0}$

Start DMA Send (SDS)

Bit 7							Bit 0
x	x	x	x	x	x	x	x

Start DMA Target Receive (SDT)

Bit 7							Bit 0
x	x	x	x	x	x	x	x

Start DMA Initiator Receive (SDI)—Mode N

Bit 7							Bit 0
x	x	x	x	x	x	x	x

X = Don't Care

Realize the Speed of Asynchronous SCSI

National Semiconductor
Application Note 575
Andrew M. Davidson

National Semiconductor's new SCSI Interface devices raise the standard of low cost SCSI.

National Semiconductor has two new low cost, high performance Small Computer Systems Interface (SCSI) devices; the DP5380 and DP8490. The DP5380 Asynchronous SCSI Interface (ASI) is pin and program compatible with the NMOS NCR5380 type device, but since it is manufactured in CMOS it offers speed and power advantages. The DP8490 Enhanced Asynchronous SCSI Interface (EASI) is pin compatible with the ASI, and program compatible until its enhanced mode is set. This mode offers features and architectural improvements which can greatly increase a system's throughput. However a major factor in favor of both devices is the improvement in their achievable data transfer rate.

Users who have replaced their NMOS 5380 type devices for National devices, in an existing application, have found improvements of between 10 and 15 percent in data transfer speeds. That is with no other changes to their application. For new designs, or performance upgrades, the gains possible can be much more dramatic.

In many applications the limiting factor in a data transfer is the DMA controller. This presented a problem to users i.e., "How fast **can** the National devices transfer data?" To determine this National designed an asynchronous DMA controller, using a programmable gate array, which can go as quickly as the SCSI device will allow it. **In these tests the National devices were up to 140% faster than the NCR5380 equivalent.**

SYSTEM OVERVIEW

A general SCSI application is shown in *Figure 1*. This shows the hardware required to interface a SCSI bus to a peripheral controller, without the devices to control the peripheral itself. The Central Processing Unit (CPU), the only "intelligent" device, controls all operations. It communicates with the SCSI bus through the EASI or ASI, which both support selection, reselection, arbitration and all other bus phases detailed in the ANS X3.131-1986 SCSI standard defined by the ANSI X3T9.2 committee. Since these devices can act as both target and initiator this could be the core of a peripheral controller or a host adaptor.

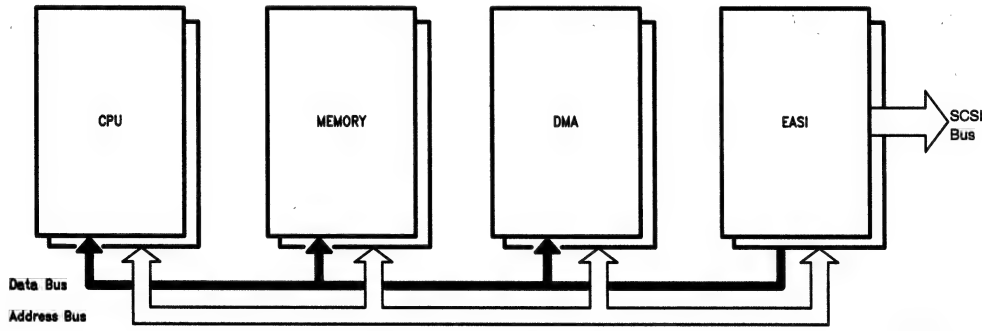
For optimum performance a DMA controller is used to transfer data between the SCSI device and memory. This memory is used as a cache for the final destination, which could be main system memory, a hard disk, floppy disk, printer, etc. By adding the appropriate interface the core shown in *Figure 1* can be adapted for the particular application.

TEST SYSTEM

To fully test the achievable DMA rate National built two boards of the form shown in *Figure 1*. The first board configures itself as an initiator, the second acts as a target. The initiator arbitrates for the SCSI bus, until successful, then selects the target and sends a Write command. This causes a DMA transfer from the initiator to the target.

The initiator arbitrates for the bus again, selects the target and sends a Read command. The data sent to the target is returned to the initiator, under DMA control, and checked. This process is constantly repeated, with the transmitted data varying between "01,FF,00,01 etc." and "FF,01,00,FF etc." This checks every data bit and varies the parity bit.

The two boards continually transmit blocks of data, alternating between Initiator Send/Target Receive and Initiator Receive/Target Send modes.



TL/F/10104–1

FIGURE 1. Core of a SCSI Controller

TEST CIRCUIT

The two boards have identical circuitry, shown in *Figure 2*. The CPU is an NSC800™ 8-bit CMOS microprocessor, which requires an octal latch to demultiplex the address and data bus. The boards configuration, initiator or target, depends on the system software stored in the EPROM. Fast static RAM is required for the cache, CMOS RAM with an access time of 45 ns was selected. A PAL® is used to decode the chip selects. The SCSI controller can be either a 5380 or a DP8490.

The DMA controller was implemented in a Xilinx Programmable Gate Array. These devices offer a quick, easily adaptable method of prototyping. The design is entered as a schematic, then converted to the gate array format by a software package. Routing is largely automatic, but complex designs require a certain amount of interaction. In a speed critical design, such as the DMA controller, manual control of routing delays is vital.

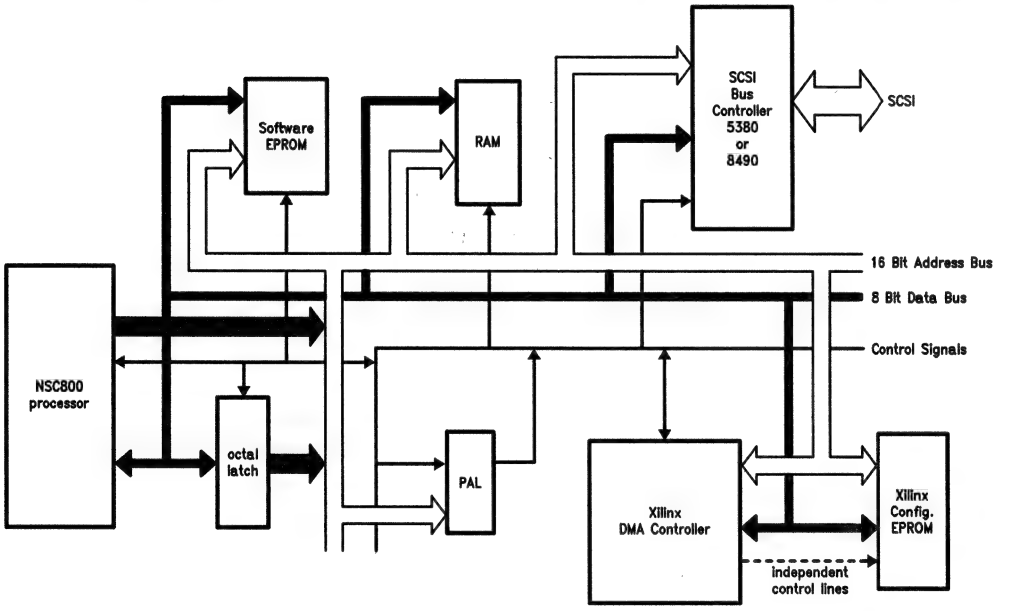
The Xilinx device can automatically load its configuration from an EPROM on power-up. This could be the same EPROM that contains the boards software, but for simplicity during prototyping a separate device was used.

DMA DESIGN

The DMA controller required was highly application specific and could therefore be relatively simple. It basically consists of a counter, a register and control logic *(Figure 3)*. The counter is loaded with the start address of the DMA block and the register with the end address. On a subsequent DMA request the controller has to take control of the microprocessor bus and transfer data until the address in the counter equals the address in the end register.

The DMA is enabled by setting a bit in the control register. The other control bit determines whether the transfer is a memory read or a memory write.

Once enabled, a subsequent DMA request (DRQ) will cause the device to issue a bus request ($\overline{BRQ}$). When it receives a bus acknowledge ($\overline{BACK}$) from the processor it asserts the counter output onto the address bus and issues a DMA acknowledge ($\overline{DACK}$). The DMA controller also takes control of the I/O and memory read and write strobes. This device only operates in block mode, so $\overline{DACK}$ is asserted throughout the transfer. The SCSI device deasserts DRQ when $\overline{DACK}$ goes active. This is used to start the first transfer. Subsequent transfers are triggered by the throttle control, READY, going high.



TL/F/10104–2

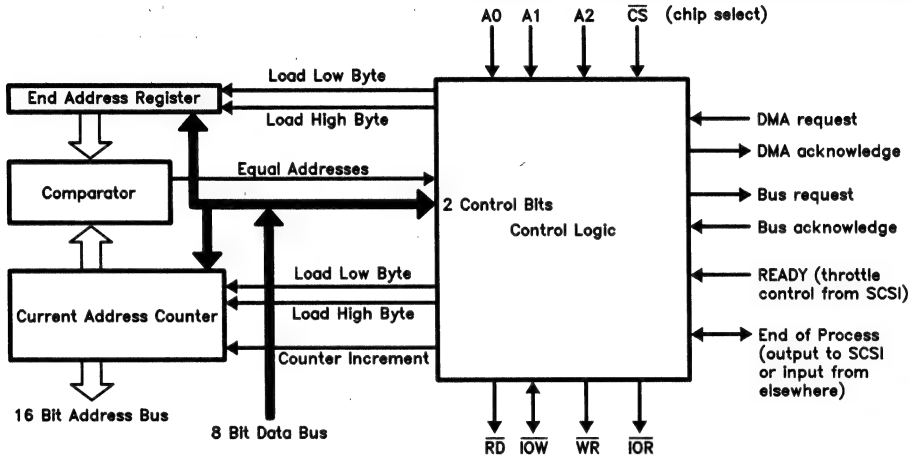
FIGURE 2. Circuit of SCSI Test Board

For a memory read, $\overline{RD}$ is held active throughout the transfer and $\overline{IOW}$ is strobed. For a memory write, both $\overline{IOR}$ and $\overline{WR}$ are strobed. The basis of the control logic is shown in *Figure 4*.

Figure 5 shows the timings of the circuit in *Figure 4*. The I/O or memory strobe pulses low for the time difference between READY going high and the STROBEIN input causing CLOCK to go high. Thus the external delay line controls the strobe signals pulse widths. The back edge of CLOCK is used to increment the address counter.

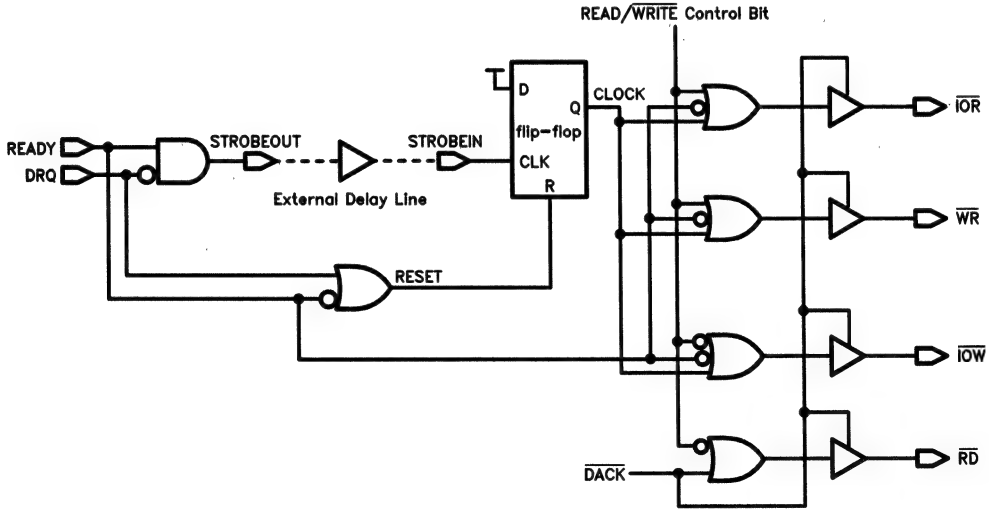
The controller continues as in *Figure 5*, until the $\overline{EOP}$ (End of Process) signal is asserted. This is either asserted externally, to prematurely end a transfer, or by the DMA controller itself, once the current address counter equals the end address register.

The speed that the DMA controller transfers data is determined by the READY line. The SCSI controller deasserts READY after an $\overline{IOW}$ or $\overline{IOR}$, then asserts it when it is ready to transfer the next byte. This high transition triggers the next transfer of the DMA controller. Varying delay line time varies the $\overline{IOR}$, $\overline{IOW}$ and $\overline{WR}$ pulse widths, so different speeds of devices can be accommodated by this controller.



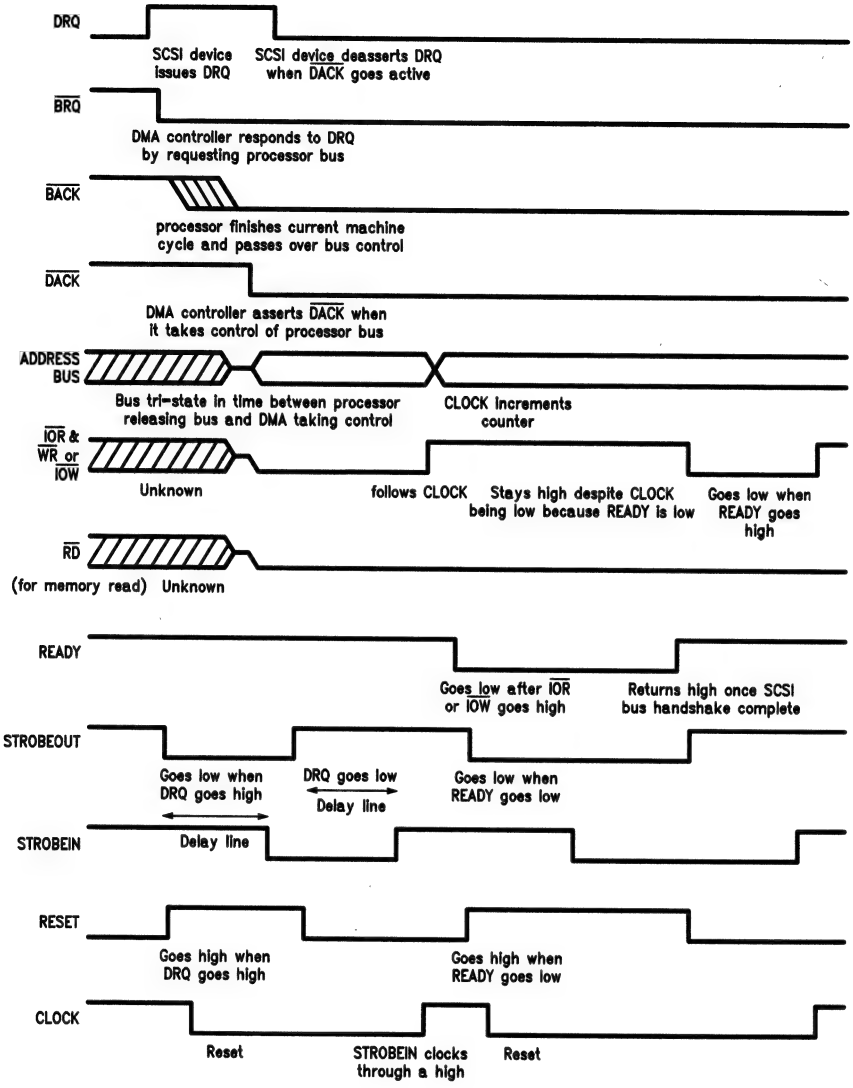
TL/F/10104–3

FIGURE 3. Block Diagram of DMA Controller



TL/F/10104–4

FIGURE 4. DMA Control Logic



TL/F/10104–5

FIGURE 5. DMA Controller Timings

BENCHMARKING

The system used for the benchmarking is shown in *Figure 6*.

The two boards were connected by three different lengths of cable; 50 cm, 3m and 6m (the maximum length of single ended cable allowed). 50-way STD IDC ribbon cable was used, wired as per SCSI specification with every second line grounded. All SCSI signal lines were terminated on both boards, with a 330Ω resistor to ground and a 220Ω resistor to V_{CC}.

A Hewlett Packard 16500A Logic Analysis system was used to monitor the control signals and measure the burst DMA rate. The 10 ns sampling rate gave sufficient accuracy to measure handshaking speeds which were all below 5 Mbytes/s.

The first byte of a transfer is delayed, as the initiator takes time to follow the phase set by the target. After this the DMA controller reaches its peak transfer rate, which is maintained until the end of the transfer. The frequency of the SCSI bus handshaking signal, REQ, was measured, to give the data transfer rate.

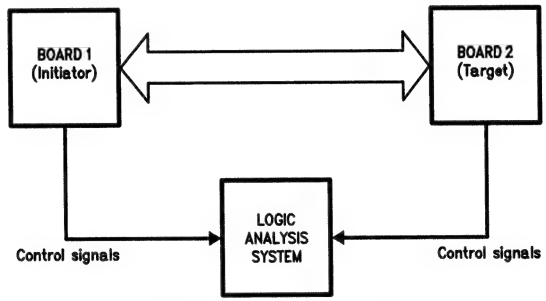
BENCHMARKED DEVICES

Initially it was planned to benchmark three manufacturers devices, that is National Semiconductor, NCR and Logic Devices.

However it was found that the Logic Devices part caused problems in the test board and would not function in conjunction with any other manufacturers device. The DMA controller uses DRQ to generate the first read and write strobes (see DMA Design section), but expects the signal to stay inactive for the rest of the transfer. The Logic Devices 5380 produces a spurious DRQ during a block mode transfer. This corrupts the read and write strobes produced by the DMA controller. The Logic Devices 5380 would not interface to other manufacturers devices because of its parity checking method. National and NCR only check the bus parity as they load their input data register. Logic Devices check parity when $\overline{ACK}$ is active, for Target Receive or when REQ is active for Initiator Receive. However, the National and NCR devices write the next byte of data before the $\overline{REQ}/\overline{ACK}$ handshake for the previous byte is complete. As parity is decoded through logic the data will be on the bus before the parity bit is valid. Logic Devices detect this as a parity error, which causes the transfer to be terminated.

The National and NCR devices were fully interchangeable.

Different Lengths of SCSI Bus



TL/F/10104–6

FIGURE 6. Benchmarking System

BENCHMARK RESULTS

The transfer rate during a data phase was measured for a sample of National and NCR devices, over three lengths of cable. Measurements were made for Initiator Send/Target Receive mode and for Initiator Receive/Target Send mode. When a NCR device was used the delay line time was increased, to allow for its slower reading and writing times. *Figures 7a, b* and *c* show the results.

These tables show the dramatic speed advantages of the National device over NCR. The National device maintains a significant advantage over all three lengths of cable. It may appear that the NCR device maintains its speed better over longer lengths of cable, but only because the added propagation delays of the cable are a smaller percentage of the transfer time.

The Initiator Receive/Target Send mode is inherently faster, which makes the specification of one DMA rate for the device a highly debatable subject, even if the length of cable for the measurement is agreed. However, customers using the SCSI device for a disk application will appreciate the fact that this mode is faster, since approximately 80% of disk accesses are reads, while running most software.

Even if only one National device is used there is a significant improvement in performance, but to fully realize the potential of the SCSI interface two National devices are required. *Figure 8* shows just how much of an improvement this is.

a. 10cm Cable

Initiator / Target	Speed in Mbytes/second	
	National	NCR
National	**3.5** **4.9**	2.6 2.7
NCR	1.8 2.3	**1.5** **2.0**

b. 3m Cable

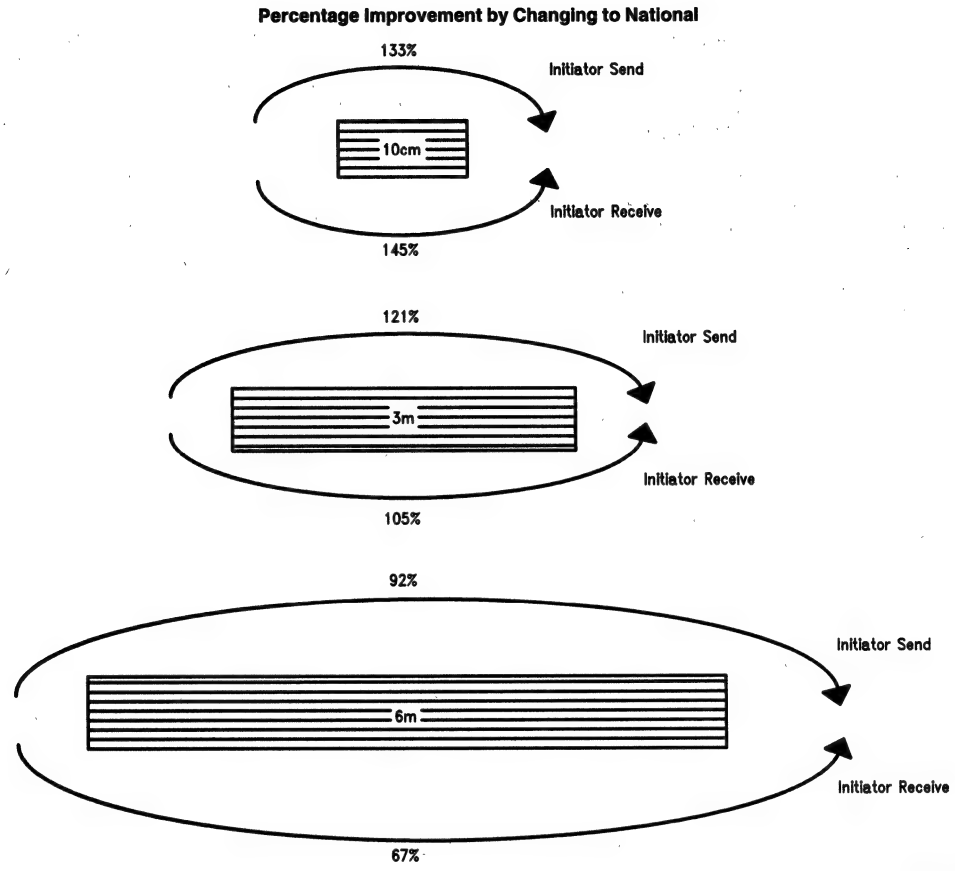
Initiator / Target	Speed in Mbytes/second	
	National	NCR
National	**3.1** **3.9**	2.3 2.5
NCR	1.6 2.2	**1.4** **1.9**

c. 6m Cable

Initiator / Target	Speed in Mbytes/second	
	National	NCR
National	**2.5** **3.0**	2.0 2.2
NCR	1.4 2.0	**1.3** **1.8**

Note: The upper value shows Initiator Send/Target Receive while the lower shows Initiator Receive/Target Send.

FIGURE 7. Measured DMA Transfer Rate



TL/F/10104–7

FIGURE 8

CONCLUSIONS

The new asynchronous SCSI interface devices from National offer an instant improvement to old designs. By changing over to National devices the user can upgrade an existing SCSI interface by upwards of 10%, with no further effort. For greater performance improvements a fast DMA controller is required, at which point the asynchronous SCSI approaches synchronous SCSI speeds.

For new designs the DP8490 EASI is the ideal choice. In addition to the fast DMA transfer rate, this device offers architectural improvements which can greatly boost a systems throughput. In a 5380 the arbitration time is dead to the system, as the device must be polled to determine when this phase has started. By interrupt driving arbitration the DP8490 EASI frees many milliseconds, or even seconds, to be used at the system designers discretion. This time may be utilized for disk cacheing, overlapped seeks, printing etc. depending on the particular application. The entire interrupt structure has been revamped, making software for this device quicker to develop and quicker to run. To further ease software overheads the troublesome bugs of the 5380 have been fixed.

By improving the speed of data transfers, lowering the power consumption and offering new additions to a familiar SCSI interface device, National have rejuvenated asynchronous SCSI.

DP8490: E.A.S.I. Does It!

National Semiconductor
Application Note 562
Andrew M. Davidson

National's Enhanced Asynchronous SCSI Interface (EASI) offers features that can yield increases in system performance over designs incorporating a 5380 type device.

When looking for a low cost Small Computer Systems Interface (SCSI) controller many users adopt the 5380 Asynchronous SCSI Interface (ASI) despite its many documented flaws and undocumented difficulties in operation, familiar to any past user. This device tended to be selected due to the lack of an alternative, but the DP8490 EASI from National Semiconductor will change this. The EASI is pin compatible with the ASI, and software compatible until an enhanced mode bit is set. This enhanced mode offers features which can increase the performance of a system currently using a 5380 type device. This is achieved through improvements to speed and architecture, while eliminating the 5380's inherent bugs.

SYSTEM OVERVIEW

The DP8490 and DP5380 support selection, reselection arbitration and all other bus phases as detailed in the ANS X3.131–1986 SCSI standard defined by the ANSI X3T9.2 committee. As the devices can act as both TARGET and INITIATOR they are suitable for any SCSI bus application. A typical application will look like *Figure 1*.

The C.P.U. controls all operations by reading and writing registers in the I/O devices; to achieve a high performance a member of the HPC16000 microcontroller family can be used. For optimum performance a DMA controller handles data transfers between the EASI and memory. The memory is used as a data cache for the final destination, which may be a hard disk, RAM disk, printer etc. *Figure 1* shows the basic system but not the hardware required to interface to such a peripheral. The EASI has high-current open-drain drivers which interface directly to the SCSI bus.

SPEED AND POWER IMPROVEMENTS

National's DP8490, and DP5380, are implemented in low voltage silicon gate microCMOS, which gives power and speed improvements over existing NMOS 5380 parts. The National devices have a maximum supply current of 4 mA, compared to the NCR5380 145 mA, and DMA rates of over 3 Mbytes/second, compared to 1.5 Mbytes/second. National's EASI device has reduced timing parameters; read access times are reduced from 130 ns to 50 ns, write data hold times, from 30 ns to 10 ns and all other parameters correspondingly improved.

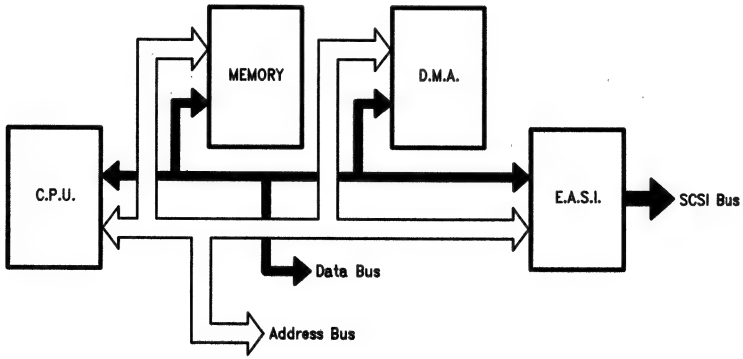
ENHANCED INTERRUPTS

The DP8490 enhanced mode interrupt structure is considerably different to that of the 5380. The source of interrupt is available by reading one register, the Interrupt Status Register (ISR). In order to find the source of interrupt in the 5380 two registers must be read. In addition all interrupts are flagged in enhanced mode. With the 5380 a selection interrupt has to be determined by the absence of other flags while the select line is active. Since selection is the means of establishing contact with other devices on the SCSI bus it is a common occurrence. The DP8490 supplies an interrupt flag for this and all other interrupts. This enhanced interrupt structure should simplify system software, thus increasing the speed of interrupt servicing. Simplified code is easier to understand, and therefore easier to adapt or repair.

All interrupts are maskable, using the Interrupt Mask Register (IMR), but interrupts cannot be lost since resetting the interrupts only resets those bits that were active on the last read of the ISR. This achieves a greater error tolerance.

INTERRUPT DRIVEN ARBITRATION

An enhancement which will have a great effect on many system's performance is the addition of an arbitration interrupt. In the 5380, arbitration is polled. On a busy SCSI bus, arbitration will take typically many milliseconds, and potentially many seconds. Therefore, time which could have been spent doing overlapped seeks, or filling a data buffer is instead used reading a 5380 register waiting for a flag to go active. The enhanced mode of the DP8490 offers interrupt driven arbitration, allowing the user to utilize this time in data cacheing, data manipulation, etc., thus increasing the system throughput. This increase in system performance can become particularly effective in low priority devices which will typically have to arbitrate for the bus the longest.



TL/F/10081–1

FIGURE 1

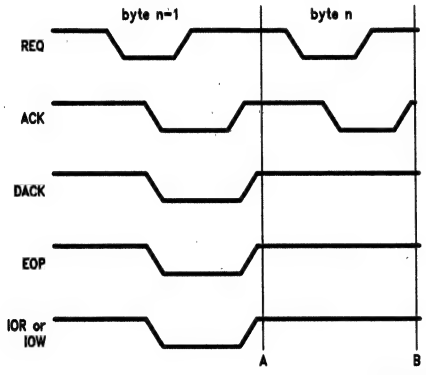
INCREASED TESTABILITY

The enhanced mode is set using a register bit which, in the 5380 initiates the 'Test Mode'. In Test Mode all output drivers on the device are disabled, so although the device can still be written, no data can be read. This makes it unsuitable for any application. In addition, DMA and interrupt requests are tri-stated, which may cause system problems.

In contrast, the enhanced mode of the DP8490 offers a loopback facility, where the SCSI drivers are disabled, and the SCSI I/O's loop back inside the EASI. This loopback test mode allows all device signals to be fully tested, including a DMA transfer. Using this, a system can perform a full self diagnostic test, without affecting the SCSI bus. Good diagnostic testing simplifies system error detection and checking.

TRUE END OF DMA DETECTION

Another interrupt improvement in the enhanced mode is seen during DMA operation. The 5380 will generate an end of DMA interrupt when it sees a concurrent EOP, DACK and IOR or IOW, even though the SCSI bus transfer may not be complete. To overcome this the user must examine REQ and ACK, to determine a true end of transfer. Consider *Figure 2*:



TL/F/10081–2

FIGURE 2

If an initiator is slow in removing the ACK, the processor could sample REQ and ACK after the EOP interrupt, at point A. At this point REQ and ACK are both inactive, although there is still one byte to transfer. The user must see REQ and ACK inactive on three successive transfers to be confident the transfer is complete (point B). Thus a great deal of time is wasted ensuring the last byte of every DMA transfer is successful. The EASI eliminates this inefficiency by detecting, and interrupting on, true end of DMA, when ACK goes inactive.

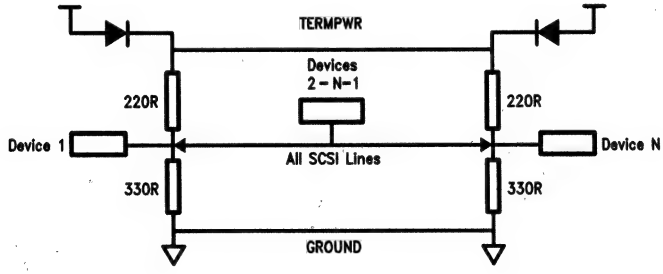
TIMING ENHANCEMENTS

Some of the major bugs in the 5380 are experienced in DMA mode. In initiator receive mode, if a REQ is received, after a valid EOP, an ACK will be generated although no valid data exists. This could be a particular problem for users who have to split a block transfer into two, perhaps to prevent crossing a page boundary. Part of the second block of data could be lost by the generation of these spurious ACK's. Another problem experienced by an initiator in DMA mode is ACK being left asserted after receipt of a valid EOP. After receiving the end of DMA interrupt the processor must 'manually' deassert ACK by writing to the relevant register.

These problems make the DMA mode in a 5380 extremely difficult to use. Extra software is required to ensure correct operation, making boards incorporating a 5380 run slower. **None of these problems exist in the enhanced mode of the DP8490.**

BUS TERMINATION

The most common SCSI bus is a 50 way ribbon cable, of up to six meters long, with SCSI devices daisy chained along it. The devices at either end of the bus should terminate all SCSI lines with a 330Ω resistor to ground and a 220Ω to power. To ensure the bus can operate correctly, even if the device at one end is not powered up, the power to these terminators can be made available on the SCSI bus. This allows the terminators to always receive power. The bus configuration is shown in *Figure 3*. Terminator power is fed through a Schottky diode to prevent a backflow of power into either of the devices.



TL/F/10081–3

FIGURE 3

Some manufacturers' CMOS devices will not work in this configuration, since when not receiving power they pull the SCSI line low. Therefore all devices on the bus must be powered up. Both National's DP5380 and DP8490 have a special input protection which makes this configuration acceptable.

EXTENDED FEATURES

The DP5380 and DP8490 are available in a standard 40 pin DIP, or in a 44 pin PLCC, which is pin compatible with existing 5380 PLCCs. The DP8490 PLCC has a microprocessor parity pin. In the enhanced mode, microprocessor bus parity checking can be enabled, with parity polarity optional. Since parity checking is optional, the PLCC DP8490 can still be used in a system where the microprocessor does not support parity. However, use of this feature will help the confirmation of data validity throughout the system.

Enhanced mode (for both types of packaging) offers other features, including programmable SCSI parity polarity. Although the specified SCSI polarity is ODD, by enabling EVEN parity bus diagnostics can be carried out. In this way the error detection and error handling capabilities of other devices on the bus can be determined.

The enhanced mode also has a general phase mismatch interrupt. The 5380 only checks phase mismatch during DMA. This means that an initiator following the phase set by a target must poll the device to determine when it changes. Only a change during DMA produces an interrupt. In the enhanced mode, an option available gives an interrupt on any phase mismatch. This allows for quicker detection of a change of phase, thus decreasing the dead time on the SCSI bus.

The wide range of new features available in the DP8490 makes this part a first choice in any new SCSI designs, or upgrades requiring increased performance. For users not wishing to make the software changes, required to utilize the enhanced features of the DP8490, the DP5380 is available, offering speed and power improvements over existing parts, at a highly competitive cost. Every user that has tried replacing existing NMOS 5380s with either the DP5380 or DP8490, at both ends of their existing application, have experienced data transfer speed improvements of between 10% and 15%. That is with no other changes!

A SCSI Printer Controller Using Either the DP8490 EASI or DP5380 ASI and Users Guide

National Semiconductor
Application Note 563
Andrew M. Davidson

The DP8490 Enhanced Asynchronous SCSI Interface and DP5380 Asynchronous SCSI Interface are CMOS devices, which offer a low cost high performance Small Computer Systems Interface. These devices are pin compatible, and software compatible until an enhanced mode bit is set in the DP8490. This enhanced mode offers many new features which can yield increases in system performance through software, in addition to the improvements in speed and power shown by both devices over existing NMOS devices.

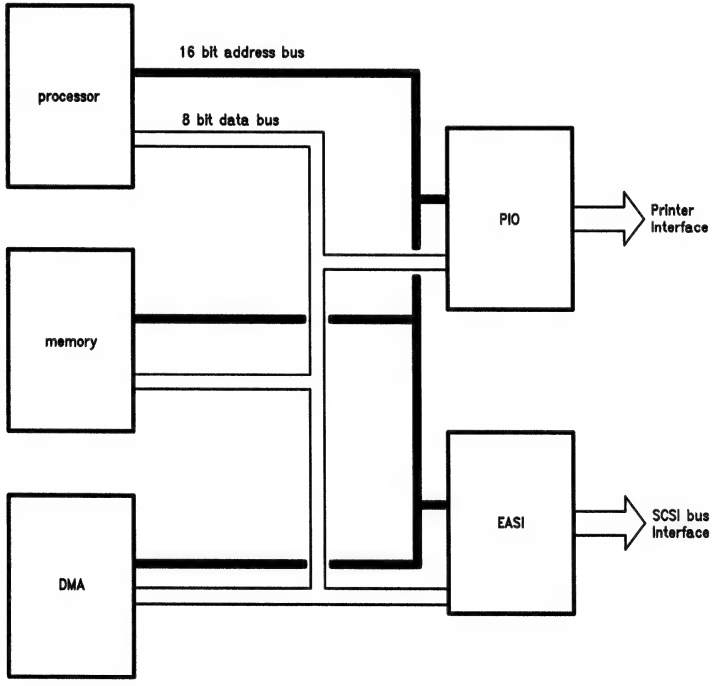
This application note shows how the hardware and software can be designed for a SCSI Printer Controller (SPC) so that it can incorporate either the DP5380 or DP8490. Since the software automatically detects which device is inserted either can be used, although the enhanced mode of the DP8490 offers a better system in terms of throughput and error tolerance. All of the software discussed is available on a floppy disk.

1.0 Introduction

The SCSI Printer Controller *(Figure 1.1)* consists of five main parts:

1. Microprocessor NSC800™
2. Printer Interface NSC831 PIO (Parallel Input/Output)
3. SCSI Interface DP8490 or DP5380
4. Memory
   CMOS EPROM NMC27C256
   CMOS Static RAM 62256
5. DMA Controller 9517 or 8237

The NSC800 is an eight bit CMOS microprocessor which is the central processing unit of the National Semiconductor



TL/F/10082–1

FIGURE 1.1

1.0 Introduction (Continued)

NSC800 microcomputer family. It is capable of addressing 64 kbytes of memory and 256 I/O devices using a multiplexed address and data bus. The instruction set is fully compatible with that of the Z80.

The NSC831 is the parallel input/output (PIO) device of the NSC800 family. This provides 20 I/O bits which can be individually programmed to be inputs or outputs. These are configured as two eight bit ports and one four bit port. The PIO is used as the interface between the microprocessor and the printer.

The DP5380 and DP8490 comply with the ANS X3.131-1986 SCSI standard as defined by the ANSI X3T9.2 committee. They can act as both Initiator and Target supporting all bus phases. Due to the on-chip high-current open-drain drivers the devices interface directly to the SCSI bus.

The 64 kbyte memory space is split between a 32 kbyte EPROM, containing the run-time software, and a 32 kbyte static RAM.

Data transfers between memory and the EASI can be controlled by the DMA controller, which supplies all read and write strobes for a transfer in either direction. This is a synchronous device which uses the 4 MHz clock output from the microprocessor.

The microprocessor is normally in control of the internal busses, giving it the ability to read or write memory or I/O devices. During a DMA transfer the DMA takes control of these busses and can pass data between the EASI and memory. As the microprocessor is the only device with 'intelligence' it must control these transfers. It commences and controls operations by setting registers in the DMA and EASI.

2.0 Hardware

2.1 DEVICES

2.1.1 Introduction

This section is intended to explain the hardware of the SPC referring to the circuit diagram given in Appendix A. This will describe the devices used and the signals generated but not the way in which they are programmed. What will be discussed is how the devices relate to each other, their relative timings, and any extra hardware required. A more exact description of the internal registers of the EASI, DMA and PIO, and their operation, will be given in the diagnostics section.

2.1.2 Microprocessor

The NSC800 is an eight bit microprocessor which multiplexes the eight bit data bus with the lower half of the address bus to create a sixteen bit address bus. An octal latch, MM74HCT373, is required to hold the address on the bus during a memory read or write, with the ALE (Address Latch Enable) output from the NSC800 strobing the latch.

Since the bus has devices which use both TTL and CMOS levels pull-up resistors are required. The $\overline{IOM}$ output signifies whether the processor is on a memory or an I/O cycle and is used along with the $\overline{RD}$ and $\overline{WR}$ outputs to strobe bus data.

This processor allows control of the bus to be passed to an I/O device, using the $\overline{BRQ}$ (bus request) input and $\overline{BACK}$ (bus acknowledge) output. On this board the DMA is the only device which may request control of the bus.

When the EASI issues a DMA request the DMA signals that it needs control of the bus by issuing a $\overline{BRQ}$. The processor acknowledges this by issuing $\overline{BACK}$, and then drives the bus and all related control signals to their high impedance state. The MM74HCT373 must TRI-STATE® its outputs; the output enable is driven by the DMA signal AEN (address enable), which it asserts when it has control of the bus. At the end of its operations the DMA releases AEN (enabling the latch outputs) and negates $\overline{BRQ}$. The processor de-asserts $\overline{BACK}$ and retakes control of the bus by enabling its outputs. To prevent spurious signals being generated when the bus is not driven the control signals must have pull-up resistors.

The NSC800 provides five hardware interrupts of which only $\overline{RSTA}$ is used. The others are tied inactive. $\overline{RSTA}$ is driven by the EASI interrupt output, with an inversion between them to allow for the difference in active levels. An interrupt on this pin causes a software reset to a particular address in memory (more details of interrupt servicing will be given in the diagnostic software section).

The system clock is generated using an 8 MHz crystal, with the frequency divided by two for use by the processor and the DMA. There is also power-on reset circuitry, which resets the processor, and causes a reset output to the other devices when power is first applied.

2.1.3 DMA Controller

The 9517 and 8237 are compatible direct memory access devices, controlled by setting internal registers. These registers are selected using a chip select, with the particular register selected by the lower nibble of the address bus, and data strobed by $\overline{IOR}$ (I/O read) or $\overline{IOW}$ (I/O write). These registers control the type of data transfer, the number of bytes transferred and the memory address the transfer is to/from.

The two modes of transfer used are single mode and block mode. In single mode the DRQ (DMA request) input and the $\overline{DACK}$ (DMA acknowledge) output are used to "handshake" every byte of data transferred. In block mode, after an initial DRQ, $\overline{DACK}$ must be active until after the last byte has been transferred. The rate of transfer is controlled by the READY input.

Another device required in conjunction with the DMA is an MM74HCT74 'D' type flip-flop. This is used to overcome the metastability problem introduced by the asynchronous READY signal driving a synchronous device. The flip-flop synchronizes the input with the system clock.

The DMA supplies all the necessary control signals to move data from memory to EASI or vice-versa. When all data has been transferred the DMA drives $\overline{EOP}$ (end of process) active. $\overline{EOP}$ can also be used as an input to the DMA, prematurely terminating any transfer. It is configured as an open-drain driver, so requires a pull-up resistor, of an advised value of 4.7 kΩ.

As in the processor the DMA has a multiplexed address and data bus, also requiring an MM74HCT373 to latch the address. In this case it is the upper byte of the address bus which is multiplexed with the data bus. The strobe signal is similar to ALE but is generated in the DMA and called ADSTB (address strobe). The TRI-STATE for this latch is driven by the inverse of AEN, since the device must drive the bus only when the DMA has control.

2.0 Hardware (Continued)

The 9517 and 8237 have four DMA channels, of which only one is used. The DRQ inputs for the other three channels are tied inactive. The RESET input causes the control registers to be cleared and the DRQ inputs to be masked.

2.1.4 EASI

The EASI is controlled by setting internal registers, written by an $\overline{IOW}$ and chip select, read by an $\overline{IOR}$ and chip select. The particular register selected is determined by the lowest three bytes of the address bus. The EASI has an eight bit microprocessor data bus, and on the PLCC part a microprocessor bus parity pin. However the NSC800 does not support parity checking so this bit is tied low. The other microprocessor controlled pin is the $\overline{RESET}$. This causes the EASI to clear all registers and therefore reset all logic.

The DRQ output and $\overline{DACK}$ input "handshake" single mode DMA, while the READY output is used to control the speed of a block mode transfer. $\overline{EOP}$ is an input which terminates DMA, and can be used to cause an interrupt. This interrupt output INT can interrupt the microprocessor if the EASI detects an error, or has completed some task.

SCSI uses an eight bit data bus with a parity bit; which must support odd parity during all bus transactions except arbitration. Other SCSI signals are the bus selection control signals $\overline{SEL}$ and $\overline{BSY}$, phase control signals $\overline{MSG}$, $\overline{C/D}$ and $\overline{I/O}$, data transfer handshakes $\overline{REQ}$ and $\overline{ACK}$ and the message flag $\overline{ATN}$. Further details on the use of these signals will be given in the software sections of this document. The SCSI reset $\overline{RST}$ is similar to a chip reset, but generates an interrupt.

The EASI interfaces directly to the SCSI bus using high-current open-drain drivers. This bus is a 50-way ribbon cable (maximum length 6.0 meters) on which all SCSI devices are daisy-chained. The devices at the end of this cable must terminate the SCSI signals i.e., a 220Ω resistor to power, and a 330Ω to ground. This power can be V_{CC} or TERMPWR (terminator power).

TERMPWR is V_{CC} fed through a Schottky barrier diode. This can be fed to pin 26 of the connector allowing the SCSI device at one end of the bus to be powered down without affecting the bus. Since the terminators are receiving power the bus can operate effectively. To make this optional TERMPWR is fed to the connector through a jumper link. The Schottky diode must be included to prevent backflow of power into the printer controller board.

Other manufacturers CMOS devices can not be used in a configuration like this, as they pull the bus low when not powered. The DP5380 and DP8490 have a special input protection to overcome this problem.

All other pins on the connector should be tied to ground, except pin 25 which must be left floating.

2.1.5 PIO

The printer controller uses a NSC831 PIO to interface between the processor and the printer. The NSC831 has 20 individually programmable I/O bits which are arranged as three ports; A, B and C. As it is part of the NSC800 family the NSC831 has a compatible multiplexed address and data bus with an ALE input, to strobe the address into an internal latch. This eight bit address can then be used to access an internal register, in conjunction with an $\overline{IOR}$ or $\overline{IOW}$ and chip select. The device has two chip selects, one of which is tied active (low), with the other coming from the address decode PAL®. The RESET input causes the three ports to become all inputs.

In this application the three ports are set up so that each port is all input or all output. Port A is an output, which drives the printer data bus through an MM74HCT241 octal buffer. Port B is an input, with the lower nibble coming from the printer outputs, driven by half an MM74HCT240 octal inverting buffer. The pull-down resistors on the $\overline{ERROR}$ and PE (paper error) inputs give a proper error signal if the printer is unconnected i.e., the same as when the printer is off-line. The upper three bytes of Port B are connected to switches, which are used to set the board's SCSI I.D. This will be further explained in the SCSI diagnostics section. Only three bytes of Port C are used, all as outputs, driven by the same device used by Port B. Bits 0 and 2 are printer outputs, while bit 3 drives a LED. This displays the 'health' of the board i.e., when the board is operational the LED is on, when non-operational it is off and when there is a known hardware error a message is 'flashed'.

The printer connector is a standard IBM® 25 way 'D' range, with all unused pins grounded, except 13.

2.2 PAL EQUATIONS

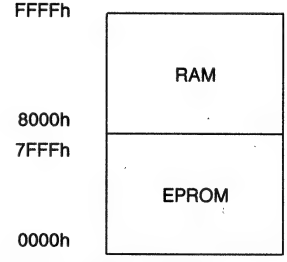
2.2.1 Introduction

PAL's are used in this board to generate chip selects, overcome potential timing problems and to invert signals for devices with different active levels. The PAL's used are National Semiconductor's PAL16L8. The equations are written in a form compatible with PLAN (Programmable Logic Analysis by National) Ver. 2.00.

2.2.2 Decode PAL

The decode PAL supplies the five main devices with their chip selects.

Memory is split into two 32k byte blocks, each of which represents one device. These devices are an EPROM, at the base of memory, and RAM in the upper half. Thus the memory map and chip select equations are as follows:



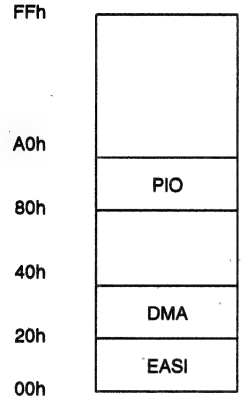
/EPROMZ = /A15*BACKZ*/IOMZ

/RAMZ = A15*BACKZ*/IOMZ + A15 * AEN

EPROMZ (the Z at the end of the name shows it is active low) can only be selected by the processor, while RAMZ is available to the DMA.

2.0 Hardware (Continued)

The DMA, PIO and EASI fit into the I/O map shown, which results in the equations following. These equations show that the devices can only be selected by the microprocessor on an I/O cycle.



/EASIZ = /A7 * /A6 * /A5 * IOMZ * BACKZ
/DMAZ = /A7 * /A6 * A5 * IOMZ * BACKZ
/PIOZ = A7 * /A6 * /A5 * IOMZ * BACKZ

The other three outputs are used as inverters:

/BRQZ = HRQ
/AENOZ = AEN
/HLDA = BACKZ

The two spare inputs are tied low.

2.2.3 Control PAL

The NSC800 uses $\overline{IOM}$ to distinguish between memory and I/O cycles. This allows $\overline{IOR}$ and $\overline{IOW}$, for processor cycles to be generated.

/IORZ = /RDZ * IOMZ
IORZ.TRST = BACKZ
/IOWZ = /WRZ * IOMZ
IOWZ.TRST = BACKZ

The second line of these equations shows that the output is TRI-STATE when the processor relinquishes control of the bus.

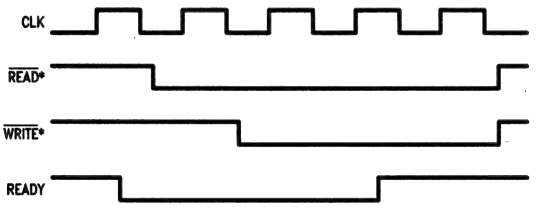
Before examining the next five equations it is important to fully understand the relative signal sequencing involved in a DMA transfer. When the transfer is initiated the EASI issues a DRQ, causing the DMA to respond with $\overline{DACK}$, and take control of the bus by asserting $\overline{BRQ}$. The processor asserts $\overline{BACK}$ and allows the bus and relevant control signals to go TRI-STATE. The DMA then asserts AEN, to show it is in control of the bus, causing the microprocessor address latch to TRI-STATE its outputs and the DMA latch to enable its outputs.

For single mode each byte transferred must have a DRQ and a $\overline{DACK}$. For block mode DRQ returns inactive after the DMA responds with a $\overline{DACK}$, but the $\overline{DACK}$ must remain active until the transfer is complete. The READY line controls the rate of block mode DMA transfer i.e., when READY is low the byte transfer is 'frozen', until READY returns high (*Figures 2.1a* and *2.1b*).

The READY signal is asynchronous but the DMA is synchronous, testing the READY signal on the negative edge of each clock. For this reason the READY input must be synchronized, using a flip-flop which updates its output on the positive edge of the clock.

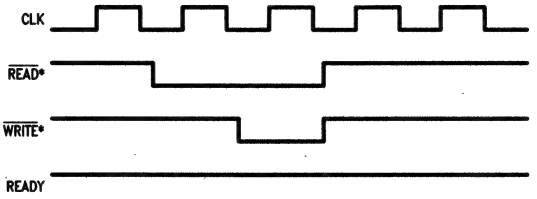
When READY is detected as being inactive on the negative edge of the clock an extra cycle is inserted to the read and write.

While data is being transferred a register keeps track of how many bytes are left. When this reaches zero an $\overline{EOP}$ signal is generated, concurrent with the last read and write, causing the EASI to set an EOP flag.



TL/F/10082–2

FIGURE 2.1a



TL/F/10082–3

FIGURE 2.1b

***Note:**

Of these read and write signals, one will refer to memory the other I/O, depending on the direction of data transfer.

2.0 Hardware (Continued)

The tranfer is now complete so $\overline{BRQ}$, AEN and $\overline{DACK}$ are all driven inactive. AEN going inactive causes the address latches to swop back over; the microprocessor latch outputs are driven, the DMA latch outputs are TRI-STATE. When the microprocessor detects that $\overline{BRQ}$ is inactive it deasserts $\overline{BACK}$ and retakes control of the bus. The microprocessor then clecks flags in the EASI to determine the success, or otherwise, of the transfer.

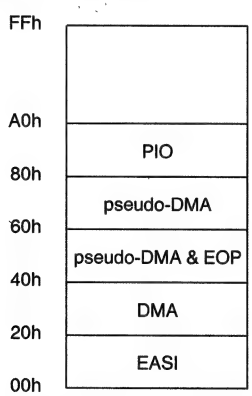
The other form of DMA available is pseudo-DMA; that is the EASI 'thinks' it is a proper DMA transfer but the processor is still in control of the bus. To do this the microprocessor must set the DMA to mask off any DRQ, initialize the EASI for single mode DMA and monitor the EASI flags for DRQ going active. At this point the processor must generate an $\overline{IOR}$ or an $\overline{IOW}$ and a $\overline{DACK}$, to properly simulate DMA. This can be done by causing an I/O read or write at a certain address, to generate a $\overline{DACK}$, and if wanted an $\overline{EOP}$. The particular address does not matter since during DMA the EASI ignores the address bus.

The following equations generate pseudo-DMA signals, while allowing the true DMA signals to operate normally:

```
/DACKOZ = /IORZ * BACKZ * /A7 * A6
        + /IOWZ * BACKZ * /A7 * A6
        + /DACKIZ
/EOP2 = /IORZ * BACKZ * /A7 * A6 * /A5
      + /IOWZ * BACKZ * /A7 * A6 * /A5
      + /EOP1 * EREADY
```

DACKIZ is the $\overline{DACK}$ from the DMA, which the output normally follows. The rest of the equation generates the pseudo-DMA $\overline{DACK}$.

$\overline{EOP2}$ is the input to the EASI, $\overline{EOP1}$ is the output from the DMA. The first two lines of this equation generate an $\overline{EOP}$ for the pseudo-DMA cycle, on the lower half of the address space that also generates the pseudo DACK. This means the final I/O map now is as follows:



The third part of the $\overline{EOP1}$ equation is to overcome a problem that occurs during a block mode DMA transfer. On the last byte the READY signal from the EASI (this will be called EREADY to distinguish it from the READY into the DMA, DREADY) may be driven low to freeze the transfer. However $\overline{DACK}$, $\overline{EOP}$ and $\overline{IOR}$ or $\overline{IOW}$ will all be active causing a valid EOP condition. This is not a problem until we consider the next equation.

```
/DREADY = /EREADY * /INT * /DACKIZ
```

This equation causes the DMA READY input to go high if any of EREADY, SCSI interrupt INT or DACKIZ go high. This overcomes a potential bus lockup, caused by an error occurring during a block mode DMA transfer. On a phase or parity error the EASI will stop the transfer and generate an interrupt. If DREADY is left inactive the DMA will keep control of the microprocessor bus. This equation holds DREADY high on interrupt, allowing the DMA to pass control of the bus back to the microprocessor.

Although this equation prevents an error it also introduces a problem in the $\overline{EOP2}$ equation. As explained previously, during the last byte of a block mode transfer although READY is low a valid EOP condition may exist, which will cause an interrupt. This interrupt will then cause DREADY to be driven high, allowing the DMA to finish the transfer, but lose the last byte since the EASI is not ready. To overcome this the $\overline{EOP2}$ equation gates $\overline{EOP1}$ with EREADY, so the EASI does not see a valid EOP condition until it is able to transfer the last byte. **This problem does not occur in MODE E since it generates a true end of DMA interrupt.**

Another problem is introduced by the DREADY equation. If an error occurs during a DMA transfer the EASI will generate an interrupt and DREADY will be forced high. This allows the DMA to 'run free', writing garbage into MEMORY, and wasting SCSI bus time. To prevent this an external $\overline{EOP}$ must be applied to the DMA on error. The next equation does this:

```
/EOP1 = /DACKIZ * /EREADY * INT
EOP1.TRST = /DACKIZ * /EREADY * INT
```

$\overline{EOP1}$ is an I/O pin which normally acts as input from the DMA, so the output is TRI-STATE unless DACKIZ and EREADY are low, with INT active. When this error condition occurs an $\overline{EOP}$ is output to the DMA, terminating the data transfer. A prerequisite of this equation working properly is the existence of the DREADY equation, since an externally applied $\overline{EOP}$ will have no effect on the DMA if READY is held low.

The next equation is also required to prevent a fault occurring during a block mode DMA transfer:

```
/EASIWRZ = /IOWZ * BACKZ
         + /IOWZ * EREADY * /BACKZ
```

When the processor is in control of the bus a straightforward $\overline{IOW}$ can write to the EASI, but if the DMA is in control EREADY must be high to allow the write. The inclusion of EREADY is required because during DMA the EASI is a flowthrough latch; an $\overline{IOW}$ passes the data from the processor bus onto the SCSI bus. Therefore if the DMA reaches its next byte before the data on the SCSI bus has been transferred, the DMA will overwrite the data. To prevent this $\overline{EASIWR}$ can only be allowed when EREADY is high.

The final PAL output is used to invert the SCSI interrupt signal, to make this active high output compatible with the processor's active low input.

```
/SCSIINTZ = INT
```

3.0 Diagnostic Software

3.1 INTRODUCTION

The diagnostic software resides at the base of RAM, with the purpose of checking and initializing the printer controller board after power up and every hard reset. It must be at location zero, since after any hardware reset the program counter is cleared. The interrupt service routine is included here, since it must exist at an exact location in memory.

3.0 Diagnostic Software (Continued)

The software uses the NSC800 instruction set (fully Z80® compatible) which makes the required low level board operations faster, and allows the exact positioning of code in memory. The assembler and linking loader are from Microtec Research.

3.2 DEVICE CHECKING AND INITIALIZATION

This section will refer to the diagnostic software PRINTER.SRC and the files PRNSYM.SRC and EASISYM.SRC which contain the constants used. A full listing of all programs described in this document are available on floppy disk.

3.2.1 Memory Check and Interrupt Servicing

An 'org' statement can be used to position this program at the base of ROM, address 0. After a jump to the ROM check, the next byte stores a label defining which version of the software is installed. On an EPROM the version number can be read from address 0002h. The upper and lower nibbles should be considered as two numbers i.e., '10' defines version 1.0.

The interrupts are disabled, since the service routines are not initialized, and the EPROM is checked; simply consisting of reading the same address twice and ensuring the same value returns both times. If this test fails the system halts since a drastic error must have occurred. If the EPROM passes the test the program jumps to the RAM check.

On interrupt the NSC800 stops before its next instruction, pushes the program counter onto the stack and loads it with a new address, depending on the highest priority interrupt channel active. Interrupt A, the only channel used, causes a jump to location 003Ch. The interrupt service routine at this location pushes all of the processor registers onto the stack, and tests for a SCSI reset. SCSI reset must be checked, as this is a special condition, causing the board to be reset by a general restart routine (this routine is in EASIO.SRC and explained in section 3.5).

If the interrupt is not a SCSI reset the software makes a call to the bottom of RAM, where a jump command should be followed by a public variable RESETA. This variable can be loaded with the starting address of the routine to service the interrupt. On return from this routine the processor registers are popped back off the stack, and program control returns from the interrupt. Since RESETA is public any external software may use it, and therefore control which routine services an interrupt. RAM must be verified before this jump table is set up.

The RAM test simply checks each BIT can contain a zero and a one, then clears every byte. Once RAM has been verified the stack pointer may be initialized, allowing call statements to be included. It should be remembered that although the interrupt routine is between the ROM and RAM tests, when the program runs it will jump straight from the ROM test to the RAM test. Since the interrupt jump table is in volatile RAM the code for a jump instruction must be written into RAM after every reset.

3.2.2 PIO Initialization

The file PRNSYM.SRC contains the port addresses for all of the I/O devices. The upper nibble contains the location the device takes within the I/O map, the lower nibble selects the register within the device to be accessed.

The PIO has five types of registers which control data transfers through the device:

1. **Data Register**—Each port has an eight bit data register containing the data passed between the PIO and the processor. These are either read or write registers, depending on whether the port bits are inputs or outputs.
2. **Data Direction Registers**—Each port also has a data direction register which controls whether each of the 20 bits is an input or an output (an input is defined by a zero, an output by a one).
3. **Mode Register**—There is one three bit mode register which selects which of the four modes the device is in. This board will always require the PIO to be in Mode 0 which is the basic I/O mode. The alternatives use Port C for handshaking.
4. **Bit Clear Register**—Each port has an eight bit, bit clear register which clears any output bit whose corresponding bit in this register is high.
5. **Bit Set Register**—These are similar to the above, but allow the output bits to be set.

The PIO is set for Mode 0, with Ports A and C outputs and Port B an input. Port A drives the printer data bus, while Ports B and C read and write the printer control signals. The four printer outputs, to the Port B input, are $\overline{ACK}$, BUSY, PE and $\overline{ERROR}$. $\overline{ERROR}$ goes low when there is no paper, the printer is off-line or an error occurs; PE goes high when the printer is out of paper; BUSY goes high to show the printer can accept data; $\overline{ACK}$ pulses low to acknowledge data after a byte has been transferred. Since BUSY and $\overline{ACK}$ are both handshake signals the user must decide which to use. This software will use BUSY.

The top three bytes of Port B are used to read in the SCSIID from a block of switches. The SCSIID is the boards identifier on the SCSI bus, used in selection and arbitration, consisting of a single active bit. This is read in as a three bit number, converted to the correct bit pattern and stored in a public variable called SCSIID.

Port C is an output, driving the printer signals $\overline{INIT}$ and $\overline{STROBE}$. A low pulse on $\overline{INIT}$ of 50 μs causes the printer to be initialized; a 500 ns low pulse on $\overline{STROBE}$ causes the printer to read the data on the bus. The highest bit of Port C drives a LED, which is on during board operation, and can display an error message by occulting a fixed code. Errors are displayed by the routine *ERROR,* a public function which displays the error number as a four bit code. It does this by occulting the LED for 1 second to show a 1, ½ a second for a 0.

Since none of the PIO registers are true read/write the device cannot be tested, only initialized.

3.2.3 DMA Test and Initialization

The DMA consists of address, word-count and control registers for four channels, of which only one is used. There follows a description of the registers used, with the addresses shown in PRNSYM.SRC.

1. **Word-Count Registers**—There are two word-count registers; a sixteen bit write only base word-count register and a sixteen bit read only current word-count register. As with all sixteen bit registers in this device the two bytes of data are accessed by two successive selections of the same address. An internal flip-flop determines which byte is read or written, with the lower byte selected first after a reset. To transfer the correct number of bytes the word-count register must be written with the number of bytes to be transferred minus one.

3.0 Diagnostic Software (Continued)

2. **Address Registers**—The address registers are also sixteen bit, and called base and current. The base address register is written with the address the transfer must start at, while the current address register contains the next address to be written or read. *The DMA is tested by writing a value to the base address and reading it back from the current address.*
3. **Control Registers**—There are three types of control register which will be discussed only in the way they are used by this software. The master clear register (DMAMCL) is the software equivalent of a hardware reset; all registers are cleared and DRQ is masked off. Masking of DRQ is controlled by the parallel mask register (DMAMSK), in this case always used to allow DRQ on Channel 0 and mask off the others. The final type of register used is the mode register (DMAMOD) which controls whether a transfer is block or single mode, memory read or memory write. Each of the four channels has a six bit mode register, all at the same address, with the bottom two bits of the data bus determining which is selected.

The DMA test program initializes the device for a one byte, block mode, memory write, to an address TSTBYT. The DMA is initialized in this particular manner for use in the EASI loopback DMA test routine.

An error in DMA test will cause error signal 0.

3.3 EASI TEST AND INITIALIZATION

3.3.1 Introduction

This document will show how the registers within the DP5380 and DP8490 are used in an application. For a full register description refer to the DP5380 and DP8490 datasheets. Register names and their locations are given in PRNSYM.SRC.

3.3.2 DP8490 and DP5380 Test

The DP5380 and DP8490 are completely pin and hardware compatible, and also software compatible until the enhanced mode bit is set in the DP8490. The DP8490 powers up in normal mode (MODE N which is software compatible with the 5380) and is in MODE N after any chip reset. The initial test is therefore the same for both devices, writing to the Mode Register 2 (EASIMR2) and reading back the data. The EASIMR2 is selected since it has the most read/write bits that do not directly affect the bus.

An error in EASI test will cause error signal 1.

For the DP5380 testing is now complete, but the DP8490 enhanced mode offers a loopback test facility, where the SCSI drivers are disabled and the SCSI I/O's looped back inside the EASI. **Using this feature the user can fully check the device, and by doing a DMA transfer fully check the board.**

At this stage it is therefore neccessary to determine which device is inserted. In the DP5380 bit 6 of the Initiator Command Register (EASIICR) selects the 'test mode', which disables all output drivers on the device, making it invisible to the system. Although the device can still be written to no data can be read back, making applications very limited. In the DP8490 this bit selects the enhanced mode (MODE E) of the device.

In MODE E addresses 0–6 access the same registers as in MODE N, but address 7 is different. Instead of accessing the Reset Parity/Interrupt (EASIRPI) and Start DMA Initiator receive (EASISDI) registers, address 7 directly accesses the Enhanced Mode Register (EASIEMR) and indirectly accesses the Interrupt Mask Register (EASIIMR) and Interrupt Status Register (EASIISR). The only other difference to registers occurs in the Target Command Register (EASITCR) where one of the previously unused bits becomes a flag (explained in section 3.3.4).

To test which device is inserted EASI test sets the enhanced/test mode bit, writes data to address 7 and reads back from the same address. If the device is a DP5380 it will be in 'test mode' and the data read will be 0FFh due to the pull-up resistors on the data bus. If it is a DP8490 the data read back will be the data written, providing the only bits set are read/write. For a DP5380 the program jumps to initialization for selection; if it is a DP8490 loopback testing follows.

3.3.3 DP5380 Initialization for Selection

The DP5380 initialization involves programming the device to respond to a selection. First, the public variable DP8490 (which other programs should treat as a constant) is set FALSE to indicate that a DP8490 is not inserted. For the DP5380 to be selected it must have the SCSIID of the device in its Select Enable Register (EASISER) and parity must be enabled. The processor must enable its interrupts and set up the jump table; *intA* is an external routine which sets the processor's interrupt mask to only allow interrupts on A, and enables the interrupts (see EASIO.SRC); *main__* is an external program which responds to a SCSI selection (further explained in the Run-time software section).

If BSY is inactive for a bus settle delay (400 ns), SEL is true and the bit on the data bus corresponding to the SCSIID is active, a SCSI selection interrupt is generated. This causes a processor interrupt, taking the program from the continuous 'jr Now' instruction to the interrupt service routine, and through the jump table to *main__*. When *main__* has finished servicing the interrupt it will return to the interrupt servicing routine and then return from the interrupt. Thus the board's 'idle' state is to continually execute the 'jr Now' instruction.

3.3.4 DP8490 Loopback Test

The loopback test mode of the DP8490 allows all signals to be fully tested, including a DMA transfer, without affecting the SCSI bus. To allow this DMA transfer loopback allows the user to drive both initiator and target signals simultaneously.

After setting the DP8490 flag true all initiator signals, and then target signals, are asserted and checked. This includes a check on the data bus by writing a test value to it and reading it back. The data bus test value has an odd number of bits, and since the specified SCSI parity is ODD, the parity bit must be inactive. One of the new features offered in MODE E is programmable SCSI parity, which is tested by ensuring the parity bit becomes active when EVEN parity is enabled, and inactive when a test value with an even number of active bits is written. Parity is then returned ODD, and the parity bit should become active.

An error in Loopback testing causes error signal 2.

The next test in loopback mode is a DMA target receive transfer to a location in memory, TSTBYT. This not only tests the EASI, but also the interrupt servicing, the DMA and memory. Although the device is in loopback the software must carry out the transfer as it would normally i.e., the bus phase must be correct, $\overline{\text{BSY}}$ must be active and the SCSI bus must be asserted. The EASIMR2 must be properly set

3.0 Diagnostic Software (Continued)

up for block mode DMA, with interrupt on $\overline{EOP}$ or parity error. Since this test is interrupt driven the interrupt jump table must be loaded with the address of the routine which will service the DMA loopback test, and the interrupts must be enabled. The DMA has already been initialized for this.

Before enabling interrupts the SCSI interrupts should be reset, which in MODE N would involve reading address 7. Since the EASIEMR is now at this address the resetting of interrupts, and the start of DMA initiator receive are initiated by writing to function bits of the EASIEMR.

When the Start DMA Target receive (EASISDT) register is written the EASI will issue a $\overline{REQ}$ to show it is ready to receive data on the SCSI bus. At this point the device at the other end of the bus would normally assert $\overline{ACK}$ to show it has data available. In this case there is no other device so the user must wait for $\overline{REQ}$ to go active and then assert $\overline{ACK}$. Thus both initiator and target signals must be asserted simultaneously. The user waits for $\overline{REQ}$ to go inactive, and deasserts $\overline{ACK}$ to show the bus transfer is complete.

The program then goes into a continuous loop awaiting a SCSI interrupt. This interrupt will occur because the EASI will have issued a DRQ, when $\overline{ACK}$ went active, and the DMA wil have transferred the last test byte written to the EASI to TSTBYT, finishing with an $\overline{EOP}$. On interrupt the program will jump to address 003Ch, where all the processor registers are pushed onto the stack, a call is made to the base of RAM, and from there it will jump to the subroutine *DIAGA*.

One of the problems with DMA in a DP5380 is that end of DMA is flagged when $\overline{DACK}$, $\overline{EOP}$ and $\overline{IOR}$ or $\overline{IOW}$ are simultaneously active; although the data may not yet have been transferred on the SCSI bus. To overcome this the software must examine the SCSI handshake signals $\overline{REQ}$ and $\overline{ACK}$, both of which must be inactive on three successive samples for a true end of DMA. This is more fully explained in the DP5380 and DP8490 datasheets. **MODE E of the DP8490 detects true end of DMA, after $\overline{ACK}$ goes inactive, before generating an interrupt.**

DIAGA responds to the interrupt after the loopback DMA by checking all the correct flags have been set. The DP5380 only uses four bits of EASITCR so MODE E uses the free bit 7 as a flag, to show true end of DMA. This is the first flag checked. Address 7 not only directly addresses the EASIEMR it also indirectly addresses the EASIIMR and EASIISR. After writing the correct code to the function bits of the EASIEMR the next access of address 7 will be to the EASIISR, if it is a read, or the EASIIMR, if it is a write. **The advantage of the EASIISR over the DP5380 registers is that all interrupt information is available in one register, and every interrupt is flagged.** To check DMA the user need only read the EASIISR and ensure that the only flag active is end of DMA. The user should note that SCSI reset causes the device to revert to MODE N, from which the EASIISR can not be read, so is not flagged.

The final EASI flag test is the 'conventional' end of DMA flag in the Bus and Status Register (EASIBSR). This flag, the interrupt flag and the flag to show no phase mismatch has occurred must be the only bits active. The final loopback DMA test is to ensure that memory location TSTBYT contains the correct data.

An error in Loopback DMA test causes error signal 3.

3.3.5 DP8490 Initialization for Selection

The DP8490 initialization is very similar to that of the 5380, even calling the same routine, *main__* , to respond to selection. However, this device stays in MODE E and uses these enhancements to only allow interrupts on selection and parity by setting the EASIIMR. Selection and parity are the only valid interrupts at this point.

At the end of the DIAGA routine program control returns to the 'jr Here' instruction, which it will continually enact until a parity or selection interrupt causes a jump to *main__*.

3.4 ERROR HANDLING

Throughout all software for this board the error handling is the same for errors considered non-recoverable, which includes all errors in diagnostics. On error the board continually displays an error number, as a four bit binary code, using the LED. The subroutine *ERROR* carries out this function.

On error register 'I' should be loaded with the error number and routine *ERROR* called. This routine then occults the LED for ½ second to display a zero, 1 second to display a one, with the LED on for ½ second between flashes. The four bits, most significant bit first, are repeatedly displayed between 2 second intervals, during which time the LED is on. The timing delays are generated using a routine *DELAY*, which gives a number of ¼ second delays, the number of delays being determined by the value in the 'I' register.

The following list shows the possible errors in diagnostics. The run-time software section contains a similar listing of its error codes.

Error 0
The DMA can not be accessed.

Error 1
The EASI can not be accessed.

All other diagnostic errors concern a DP8490

Error 2
An error has occurred in the assertion of SCSI signals in loopback test mode.

Error 3
An error has occurred during the loopback DMA transfer.

3.5 EASIO.SRC

This file contains public assembly language routines, which can be called by either the diagnostics or run-time software, to implement low level commands. As these routines may need to be called by routines written in 'C' any variables passed to the routines are passed in the 'hl' register pair, then in 'de', then 'bc' and then on the stack. *ERROR* and *DELAY* both use the 'I' register to pass a variable. Data passed back to the calling routine is returned in the accumulator.

Functions *'read'* and *'write'* implement general purpose I/O register accesses, while *'dmaread'* and *'dmawrite'* handle the special case of the 16-bit DMA registers. These require two accesses of the same address. *'intA'* initializes the processor interrupt mask, with a 'pseudo' I/O write, which selects a register internal to the processor, setting the mask to only allow interrupts on RSTA. This function also enables the interrupts, which can be done by *'eni'*, with *'dsi'* disabling interrupts.

'IDtest' is used by the run-time software, during selection, to read the number of bits active on the bus. This routine checks the number of high bits in the byte passed in the 'I' register, returning the value in the accumulator.

3.0 Diagnostic Software (Continued)

Function *'restrt'* is a general purpose reset routine, which can be called on an error condition. This causes every device on the board to be reset, and the diagnostics to be rerun, thus clearing all memory and reinitializing the stack.

4.0 Run-Time Software

4.1 INTRODUCTION

Following the diagnostic software must be a program which will control all SCSI bus transfers, beginning with selection. This program is written in 'C', using a PARAGON 'C' cross compiler for an NSC800. The relevant files for this section are SCSI.C, COMMAND.LIB, PROCESS.LIB, DMA.LIB, PRINT.LIB, ARBITRAT.LIB, SYM.H, CONST.H and COMMANDS.H, all of which are on the supplied floppy disk.

SCSI.C contains the main program, called *'main'* by 'C' and *'main_'* by assembler. The '.LIB' files contain the functions called by *main* and the '.H' files contain the constant values used by all of these files.

4.2 MANDATORY PHASES

This section outlines all the bus phases and transfers which a target must support. It would be perfectly legal for a target to respond to a selection, fetch a command block from the initiator, and then returns status and message bytes before releasing the bus. Although no process would be actuated this would be a legal sucession of events.

4.2.1 Selection Response

main() is the program jumped to when an interrupt is generated as the program circles the continuous loop at the end of diagnostics. This routine should only be entered after a selection interrupt, if any other interrupt is active it is considered an error. The function *select()* checks an interrupt to ensure it is valid.

The type of device installed is checked by reading the 'DP8490' flag, and this determines how the interrupt is verified. In a DP5380 a selection interrupt is determined by the absence of any other interrupt, with $\overline{SEL}$ active. The user must check the EASIBSR to ensure that no error flags are active, only the INT and PHSM (phase match) bits.

Any other flag will cause error number 4 to be displayed.

After reading the EASIBSR and finding no interrupt flags the user knows the interrupt should be a selection. The only other unflagged interrupt is a reset, which would have been handled by the low level interrupt service routine. Therefore the EASICSB register must also be read, and if $\overline{SEL}$ is inactive an error condition exists.

This causes error number 5 to be displayed.

In MODE E the user need only read the EASIISR to determine which interrupt is active, including selection. All interrupts, other than parity and selection, should be masked off, so any error concerns only these two flags.

A SCSI parity error is displayed as error number 8, if the selection flag is not active error number 9 is displayed.

The common tests for MODES E and N both concern the EASICSD; error 6 shows that the correct SCSIID bit was not active on the bus; error 7 shows there was more than two bits active on the bus. During selection an initiator is only allowed to assert two bits on the bus, its own ID and the target ID.

The target must assert $\overline{BSY}$ to show it has recognized the selection, then when the initiator deasserts $\overline{SEL}$ the selection phase is complete.

4.2.2 Command Phase

A selection phase is followed by a command phase, where the target reads a command block from the initiator. This command block specifies the actions the initiator requires the target to execute, plus the length of any data transfers requested. This board only allows six byte command blocks, which are transferred into the target by function *fetch_cmd()*.

The command block is transferred using programmed I/O; that is each byte is individually handshaken under processor control. The bus phase must be Command Out (out and in always refer to the initiator, so Command Out is a command block sent by the initiator), bus phase being set in the EASITCR. To transfer a byte (see *Figure 4.1*) the user must assert $\overline{REQ}$, then wait for $\overline{ACK}$ to go active to show data is available. On $\overline{ACK}$ the target can read the data on the bus, then deassert $\overline{REQ}$ to show it has received the data. When the initiator deasserts $\overline{ACK}$ the byte transfer is complete.

4.2.3 Status Phase

The target must send a status byte to the initiator during the status phase, at the termination of each command. A list of status codes is given in COMMANDS.H. *status()* is the function which enters a Status In phase, and transfers the code.

The EASITCR must be written with the Status In phase and the EASIODR with the status code. This code should be asserted onto the bus, remembering to keep $\overline{BSY}$ asserted, and if neccessary the MODE E bit. $\overline{REQ}$ is then asserted to show data is available. The initiator should assert $\overline{ACK}$ when it has read the data, allowing the target to deassert $\overline{REQ}$ and take the data off the bus. The transfer is complete when the initiator deasserts $\overline{ACK}$ (See *Figure 4.2*).

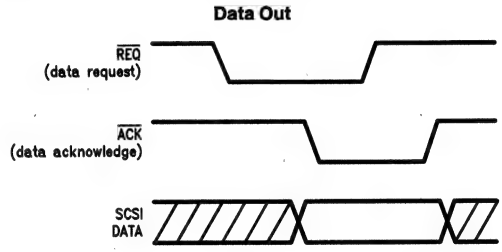


FIGURE 4.1

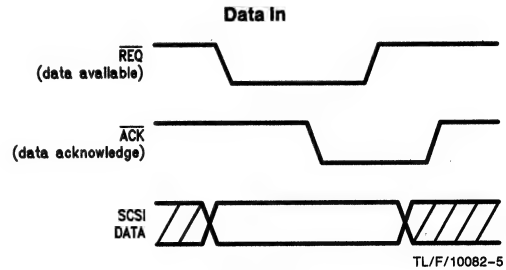


FIGURE 4.2

4.0 Run-Time Software (Continued)

4.2.4 Message Phases

A selection must be terminated by the target entering the Message In phase, and sending a relevant message code (as listed in COMMANDS.H). The target follows the Status phase with the Message In phase, usually to send the COMMAND COMPLETE message. This indicates valid status exists, so the target can release $\overline{BSY}$ to free the bus. *messin()* enters the Message In phase and sends the message using programmed I/O, as in *status()*.

The only message which must be supported is COMMAND COMPLETE. If a device on the bus supports other messages it indicates this by responding to or asserting $\overline{ATN}$. An initiator asserts $\overline{ATN}$ if it has a message for the target. It is common for an initiator to assert $\overline{ATN}$ during selection and, if the target responds by entering a Message Out phase, it sends the IDENTIFY message. This establishes whether the target can respond to a greater set of messages, and whether the initiator supports disconnection. It shows this by setting bit 6 of the message. In a disk controller IDENTIFY would also establish the path by including a Logical Unit Number (LUN), but the printer controller uses all LUN's at this SCSIID.

getmes() enters a Message Out phase and fetches a message using programmed I/O. Although this software only supports single byte messages it must be prepared to accept messages from the initiator of up to the maximum length, 256 bytes. If an initiator wishes to send further bytes after the first it must keep $\overline{ATN}$ asserted, only deasserting after the final byte has been transferred. *getmes()* will handshake up to 256 bytes, after which if $\overline{ATN}$ is asserted it will be considered an error. Only the first byte is used in determining the message sent.

The use of messages during disconnection, arbitration, reselection and in error handling will be discussed in those sections of this document.

4.2.5 Command Termination

After sending status and message codes to the initiator, the target should then release $\overline{BSY}$ to free the SCSI bus. However, before allowing a bus free phase the target must initialize itself for the next selection, as in function *reset()*.

The interrupt jump table is loaded with *main()* and the EASISER with the SCSIID. For a MODE E device the interrupt mask is set. For both types of device the interrupts must be reset, and $\overline{BSY}$ deasserted. Interrupts are disabled at the start of this function, so must be enabled after calling *reset()*.

4.3 COMMAND PROCESSING

After the command block has been fetched the command has to be determined and executed. The printer command set is shown in COMMANDS.H.

process_cmd() reads the first byte of the command block which contains the operation code. This determines what actions are taken. If a command has been sent which this software does not recognize, a status of CHECK CONDITION is returned, with sense set to ILLEGAL REQUEST.

4.3.1 Test Unit Ready

TEST UNIT READY will be sent by an initiator before a print to ensure the printer is on, on-line, has paper and is not in an error condition. On this command the function *ck_printer()* is called to read the printer signals through Port B of the PIO and check for errors.

If the error line is not high the printer is operational, so the status is GOOD, and the sense is set to NO SENSE (no error). If there is an error the paper error line must be checked. If this is low (active low input) the printer is out of paper and status of CHECK CONDITION is returned, with sense set to MEDIUM ERROR. If it is not a paper error the printer is assumed to be off or off-line, so status is CHECK CONDITION with sense UNIT ATTENTION.

4.3.2 Request Sense

An initiator will send a REQUEST SENSE command after the target has returned a status of CHECK CONDITION. The sense data is sent to the initiator in an effort to understand an error condition, and if possible recover from it. Byte 4 of this command block contains the length of an extended sense message, which must not be greater than 4, or sense is set to ILLEGAL REQUEST and CHECK CONDITION status returned. This software only supports four bytes of sense data.

Sense data is sent to the initiator using single mode DMA. The DMA is initialized by *send_sense()*; it is reset by a master clear, the mode set, the DMA mask written and the address and word-count loaded. The function *single_dma_in()* (explained in section 4.4.1) enters the correct phase (Data In) and transfers the data.

4.3.3 Reserve Unit

In a multi-initiator system a printer controller must be reserved before a print commences, or the data from two different initiators may be mixed. This command stores the initiator's ID in a variable called 'reserved', and on subsequent selections only this initiator may execute commands.

Any initiator wishing to reserve the unit must put its own ID on the bus during selection, along with the target's ID, so this software knows which initiator is reserving the unit. If the initiator's ID is not on the bus it can not reserve the unit, and since this is a prerequisite of printing, it cannot use the printer.

If an initiator attempts to reserve this unit without making its ID available sense is set to ILLEGAL REQUEST and status of CHECK CONDITION returned. If another device attempts to reserve the board when it is already reserved status returned is RESERVATION CONFLICT.

4.3.4 Release Unit

This is the reciprocal command to the previous, freeing the printer for other initiators after a print has been completed. If an initiator other than the reserver attempts this command a status of RESERVATION CONFLICT is returned; if this is attempted when there is no reservation current the returned status is CHECK CONDITION with sense set to ILLEGAL REQUEST.

4.0 Run-Time Software (Continued)

4.3.5 Print

Since transferring data to a printer is a very slow process, typical Epson Fx range 80 cps–160 cps, the print command transfers the data to a buffer, leaving it to be printed later. Bytes 2, 3 and 4 of the command block contain the data length, byte 2 the most significant. The use of three bytes to define the size means that in theory block transfers of up to 16 MB are allowed. This software could support blocks of up to 64 kB, bytes 3 and 4 are read, but blocks are limited to a size BUFFLIM, which will be determined in section 4.4.3.

Possible errors in a print occur when the unit is not reserved, the data length is set to zero, or the block size is too large. The block size is too large if byte 2 of the command block is active or if the data length is greater than BUFFLIM. In response to an error the transfer is cancelled and status of CHECK CONDITION is returned with sense set to ILLEGAL REQUEST.

The data is not transferred unless sense is set to NO SENSE. If an error has occurred in the printer sense will have been set to indicate the error source. The error condition must be rectified and TEST UNIT READY sent to reset the sense data.

print_cmd( ) is the function which transfers data into a buffer using block mode DMA. The print buffer is a circular queue, allowing the user to take data off the front and put data on the rear (see *Figure 4.3*).

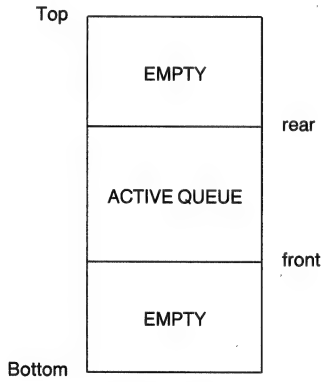


FIGURE 4.3

front points to the next byte to be printed, rear points to the next available byte for entering data. When front equals rear the queue is empty, when rear is one less than front the queue is full.

The buffer limits are called top and bottom. When either the front or rear pointers reach the top the next increment takes them to the bottom. Thus the queue may look like *Figure 4.4.*

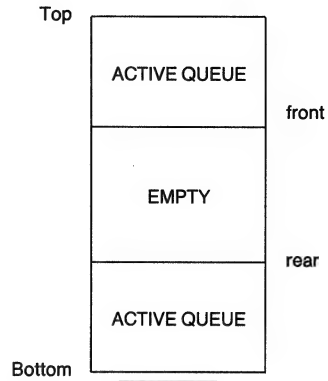


FIGURE 4.4

When *print_cmd( )* is first called after a reset it must set up the queue, defining top and bottom, and setting front equal to rear equal to bottom. Before any transfer the data length must be checked to ensure that the DMA will not take rear past top. If it would the data must be transferred in two blocks; one to the top of the queue, and one starting at the bottom of the queue. This routine should not be called unless there is sufficient free space for the size of transfer. Function *dma_data( )* sets up the EASI and DMA for a block mode transfer, enters the Data Out phase and calls *dma( )* to handle the transfer. This will be more fully explained in section 4.4.

4.3.6 Flush Buffer

The purpose of this command is to allow an initiator to terminate an unwanted print. An initialization pulse is sent to the printer, in case it has its own buffer, and the SPC buffer is cleared. This is done by setting front equal to rear equal to bottom.

4.4 DATA TRANSFERS

This section is concerted with the way in which software controls DMA transfers, both block and single mode, and how the print data block length is determined.

4.4.1 Single Mode DMA

The four bytes of sense data are sent to the initiator using single mode DMA. *send_sense( )* sets up the DMA to transfer four bytes of data from the sense buffer to the EASI, leaving function *single_dma_in( )* to complete the transfer.

The phase, which in this case is Data In, is determined by the calling function, and the EASIMR2 initialized for a single mode transfer. Parity checking is enabled with interrupt on $\overline{\text{EOP}}$ or parity error. The data bus is asserted and the routine *dma( )* is called to handle the transfer.

Since both single and block mode DMA transfers, either in or out of the initiator, have large portions of code that are common, a function can be written for general purpose DMA handling. This function, *dma( )*, sets up the interrupt response and makes a write to the register which initiates the required type of transfer, which could be target receive or target send etc. *dma( )* checks the success of the transfer when the transfer is complete.

4.0 Run-Time Software (Continued)

In MODE E the interrupt mask can be set to only allow parity, $\overline{EOP}$ and DMA phase mismatch interrupts. The interrupt jump table is loaded with the starting address of a service routine, which sets a flag to show an interrupt has occurred. The program can then sit in a loop waiting for this flag to go active.

After the interrupt the cause has to be checked to ensure the transfer was successful, but this checking depends on the device installed. For a DP8490 the user can read the EASIISR and if any flag other than end of DMA is active an error has occurred. On such an error, status is set to CHECK CONDITION with a sense of ABORTED COMMAND. The interrupt mask can be reset, masking off end of DMA and DMA phase mismatch, and the DMA mode in the EASIMR2 disabled. The interrupt service routine is set to jump to the general interrupt handler *gen_int()*. The interrupt must then be reset and the processor's interrupts enabled.

In MODE E true end of DMA is detected, but in MODE N the end of DMA interrupt is generated when $\overline{EOP}$, $\overline{DACK}$ and $\overline{IOR}$ or $\overline{IOW}$ are active concurrently. True end of DMA, after the transfer is complete, must be detected by software. $\overline{REQ}$ and $\overline{ACK}$ must both be inactive on three successive samples for true end of DMA. **This additional code makes the MODE N DMA routine slower**. After detecting this the user can check the end of DMA flag is active in the EASIBSR, giving status CHECK CONDITION and setting sense ABORTED COMMAND if it is not. The DMA bit is then reset and *gen_int()* called to ensure there was no parity or phase errors, and to reset the interrupt service routine.

On return to the function *single_dma_in()* it will deassert the SCSI bus and return to the calling routine.

4.4.2 Block Mode DMA

blk_dma_in() is the equivalent routine to *single_dma_in()* initiating a block mode transfer to the initiator. *send_sense()* could call this routine, after setting the DMA for block mode, and transfer the data using block mode, with no difference to the other software. The only difference between these two routines is the setting of the EASIMR2 BLK mode bit.

print_cmd() is used to transfer the data to be printed from the initiator. This function checks that the transfer will not exceed the constraints of the queue and calls *dma_data()* to initiate a block mode transfer. *dma_data()* sets the EASI and DMA registers for a block mode target receive, with the data size set by *print_cmd()* and the destination starting address equal to rear. *dma()* is again used to handle the interrupts.

4.4.3 Determination of Data Block Length

The block of data to be transferred for a print has been specified as a maximum length of BUFFLIM, but the value of this constant has to be evaluated. The two main considerations in this are the latencies of the bus and the SPC.

Since this board is a printer controller it will be of a very low priority. Thus the device should not hold the bus for too great a time per transfer, as this would slow down initiator's accesses of a high priority peripheral. The second consideration is the time that the target takes from selection to entering a data phase, the data block must take longer than this to transfer, or the command is inefficient. Measurements taken on various models of personal computers showed that, with an Advanced Storage Concepts ASC-88™ SCSI Host Adaptor, the time taken from the initiator asserting $\overline{SEL}$ to the target commencing the data transfer is approximately 3 ms.

The block mode DMA rate was measured as between 200 kB/s and 500 kB/s limited by the DMA in the PC. A block length of 2 kB was selected, since this would take between 4 ms and 10 ms to transfer. Thus the data transfer time is greater than the selection to data phase time, but the overall time on the bus in not too long.

4.5 PRINTING DATA

Due to the inherent slowness of printers data can not be printed while the controller is in command of the SCSI bus. To prevent tying up the bus data is stored in a buffer and printed after $\overline{BSY}$ has been released. The function *printit()* handles the transfer of data to the printer.

printit() checks the printer is not in an error condition, and transfers the byte of data at address front to the printer. The data is transferred through Port A of the PIO with BUSY and $\overline{STROBE}$ used to handshake the data. BUSY must be high and $\overline{STROBE}$ pulsed low a minimum of 0.5 μs for the printer to accept data. front is incremented, to point to the next byte to be transferred. To print out the queue *printit()* is called until front equals rear.

If the function does detect an error it sets sense to the appropriate value, and implements any outstanding reconnection. The printer is continually polled to determine when it comes back on-line, at which time the print continues.

While the board is printing it must still be available for selection. An initiator can reset every device on the SCSI bus if a target does not respond to a selection within a Selection Timeout, normally 250 ms. The interrupt service routine has been set to jump to *main()*, but the program is currently in *main()*, so this function must be re-entrant.

main() is written in such a way that it will only process commands if it is unreserved and not printing, or reserved by the current initiator but not printing. If the board is reserved, but not by the current initiator, a status of RESERVATION CONFLICT is returned. If the board is printing, and the reserving initiator attempts to send it another command a status of BUSY can be returned.

However, the controller has a 30 kB print buffer, of which only 2 kB would be used at a time. It would be much more efficient to continue executing commands until the buffer is full. One method of achieving this can be seen in *outbuf()*. In this function a flag, called next, is set if the free space in the queue is greater than BUFFLIM. If a selection occurs while a print is on, this flag is checked and the command processed if it is active. If it is inactive a status of BUSY is returned. This method has the disadvantage that the initiator must continually poll the SPC to determine when it is ready to accept data. This 'loads' the SCSI bus and slows down the print, since the SPC will be responding to selection. It also restricts other device's use of the bus. The alternative method of utilizing the buffer space is to use disconnection and reconnection.

4.0 Run-Time Software (Continued)

4.6 DISCONNECTION AND RECONNECTION

4.6.1 Disconnection

If an initiator sends the IDENTIFY message to the target, it can indicate that it supports disconnection by setting bit 6 of this message. If this bit is not set, or the message not sent, the initiator is assumed not to support disconnection, and no disconnection is attempted. This software uses disconnection in two places; 1) the board is reserved, currently printing and selected by the reserving initiator; 2) the board has been released, but not finished printing.

If the board is busy printing it can be advantageous to disconnect from the reserved initiator after reading in the command block, and then reconnect having ensured that the command can be implemented i.e., there is enough free space in the queue. For any other initiator the response is the same as before, the status of RESERVATION CONFLICT is returned until the unit is released. After this the controller will disconnect from any initiator attempting to select it, until it has enough free space in its buffer.

disconnect() is the routine which carries out the disconnection procedure. This can only take place after the command phase and before the data phase, with an initiator that sent the IDENTIFY message with the disconnection bit set. The target follows the Command Out phase with a Message In phase and sends the DISCONNECT message, after which it can drop $\overline{BSY}$, to release the bus. If the DISCONNECT message is not sent the initiator will treat the deasserting of $\overline{BSY}$ as an illegal termination of command. Before releasing $\overline{BSY}$ *exchange()* is called to store the ID of the initiator and the command it wants to execute. *reset()* is again used to initialize the interrupts and release $\overline{BSY}$.

disconnect() stores in a variable, recon_data, the amount of print buffer which will be required to execute the command. For a print, this depends on the length of data to be transferred, for any other command the amount is zero. recon_data is used to determine when the target can reconnect to the initiator i.e., once there is enough space free in the buffer.

After disconnection the program will return to printing, at which point it may be selected by another initiator. As it can only disconnect from one initiator at a time it must return a status of BUSY.

In *outbuf()* it can be seen that reconnection takes place when the free space in the print buffer is greater than the number of bytes to be transferred. *reconnect()* calls the functions which action the correct reconnection procedure, beginning with arbitration.

4.6.2 Arbitration

Arbitration requires two functions, one for MODE E another for MODE N, due to the fundamental difference in their methods of arbitration i.e., **MODE E is interrupt driven, MODE N is polled**. These two functions, *Narbitrate()* and *Earbitrate()*, carry out the same basic operation, arbitrate for the bus until successful, but do so in distinctly different ways.

To arbitrate for the bus a device must wait for a Bus Free Phase, when $\overline{BSY}$ is continuously inactive for 400 ns with $\overline{SEL}$ inactive. After a Bus Free Delay of 800 ns the SCSI device should assert $\overline{BSY}$, and assert its SCSIID bit on the bus. After a further 2.2 μs Arbitration Delay the data bus should be examined, and the device with the highest priority SCSI ID bit asserted wins arbitration.

In MODE N the EASIODR should be written with the SCSIID and the arbitration bit in the EASIMR2 set. The interrupts must be initialized to cause a jump to routine *servn()*. The EASI will wait for a Bus Free Phase then, after a Bus Free Delay, it will assert $\overline{BSY}$, and put the contents of the EASIODR onto the bus. The user must poll the AIP bit of the EASIICR to determine when arbitration has begun and check the LA bit to ensure that arbitration has not been lost. The LA bit is set if another initiator asserts $\overline{SEL}$ during arbitration. If AIP is active and LA inactive the user must examine the EASICSD to determine whether it is the highest priority device arbitrating. If arbitration is lost the arbitration bit in the EASIMR2 must be reset and the whole procedure begun again. The device shows it has won arbitration by asserting $\overline{SEL}$. An interrupt during MODE N arbitration is treated as a selection, so *servn()* enables parity, since there is no parity checking during arbitration, and calls *main()*.

In MODE E arbitration is interrupt driven, allowing the board to continue printing while waiting to arbitrate. For a busy SCSI bus typically many milliseconds, and potentially many seconds, can be taken up arbitrating. In MODE E this time can be utilized, thus increasing the system throughput. For this application the time gained is used to continue printing, in other applications, such as a disk controller, it could be used in data cacheing, overlapped seeks, etc.

In MODE E arbitration the EASIODR must be written with the SCSIID and parity checking cancelled in the EASIMR2. The interrupts are set to allow selection and arbitration interrupts, with the jump table loaded with *serva()*. This sets a flag to show an interrupt has occurred. The enhanced arbitration is initiated, with a write to the arbitration bit of the EASIEMR. This causes the EASI to wait for a Bus Free Phase; delay a Bus Free Delay; assert $\overline{BSY}$ and the EASIODR; delay an Arbitration Delay then interrupt the processor. The interrupt causes the flag to be set that shows the state of arbitration should be examined. While waiting for this interrupt the printer carries on printing out data, until front equals rear.

After the interrupt is detected the EASIISR can be read to determine the cause. If it is not an arbitration interrupt parity checking is enabled and the interrupt treated as a selection, by calling *main()*. If the interrupt is signalling the commencement of arbitration, the procedure is the same as in MODE N, with the LA bit of the EASIICR being examined, and priority determined by reading the EASICSD. If arbitration is lost, interrupts must be reset, the arbitration bit in the EASIEMR reset, and the whole procedure begun again. If arbitration is successful the user asserts $\overline{SEL}$.

Function *printarb()* is used to print data during arbitration in the same way as *printit()* does normally. The difference here is that the arbitration interrupt should be serviced as quickly as possible.

This routine must be left if an interrupt occurs. Instead of waiting for the printer to come on-line if an error occurs this routine simply does not send the character.

4.0 Run-Time Software (Continued)

4.6.3 Reselection

When arbitration has been won the disconnected initiators ID and command block must be restored, and the initiator reselected. This reselection is carried out by function *reselect( )*.

The EASISER is cleared to stop it responding to the reselection and the initiator ID bit written into the EASIODR along with the SCSIID. I/O must be asserted to show this is a reselection. The EASIODR is asserted onto the bus and the relevant arbitration bit, depending on MODE reset. This deasserts $\overline{BSY}$. Parity checking is enabled, and the board waits a selection timeout delay of 250 ms for the initiator to respond, by asserting $\overline{BSY}$. If it does not, $\overline{RST}$ is asserted, resetting the whole SCSI bus.

If the initiator does respond, the target must assert $\overline{BSY}$, then deassert $\overline{SEL}$ and take the EASIODR off the bus. For MODE E the interrupt mask can be set, and the arbitration interrupt reset. Reselection is now complete.

4.6.4 Reconnection

After *reselect( )* the IDENTIFY message is sent by the target with the disconnect bit set. The command that was previously sent is processed, if this message is received successfully. After processing the command status and the COMMAND COMPLETE message are returned. *reset( )* is again used to set the interrupts and release $\overline{BSY}$.

If a print was not current at the time of reconnection, and the reconnected command was PRINT, this is handled within *reconnect( )*. As long as the status is GOOD the print is carried out. However, *reconnect( )* will be mostly called from *outbuf( )*, when the queue has enough free space to process the outstanding command. Any reconnection missed by *outbuf( )* is captured in *main( )*.

4.7 ERROR HANDLING

Throughout this document the handling of errors has been discussed as the errors arose, but there are some remaining to be discussed. These are phase or parity errors during command transfers and message errors.

4.7.1 General Error Handling

While this board is connected to the SCSI bus, with $\overline{BSY}$ active, *gen_int( )* is generally used to handle interrupts. This routine responds to an unexpected interrupt by checking the parity and phase flags, in the EASIBSR for MODE N, the EASIISR for MODE E. It then resets the interrupts, before setting appropriate flags. Sense is set to ABORTED COMMAND with a status of CHECK CONDITION if an error has occurred.

After calling *select( )*, to respond to selection, *main( )* then calls function *set_up( )*, which sets the mask and jump table to respond to an interrupt. This is also the routine which checks to see if $\overline{ATN}$ is asserted. If it is *getmes( )* is called to enter the Message Out phase and fetch the IDENTIFY message. *messout( )* processes the message. This is where the target determines if the initiator supports disconnection.

If an error occurs during *select( )* or *set-up( )*, detected by *gen_int( )*, the board sends relevant status and a message of COMMAND COMPLETE, before releasing the bus. No attempt is made to recover from the error. Similarly, if a phase error occurs during the command phase, status and sense are returned. However, if a parity error occurs during a command phase, the target can attempt a recovery by sending the RESTORE POINTERS message. This message instructs the initiator to reset the command pointer to the beginning of the command block, allowing the target to re-enter the Command Out phase and re-transfer the command block. If there is a parity error again, status is returned, along with the COMMAND COMPLETE message. If the second transfer is successful, the command execution continues as normal.

4.7.2 Message Errors

If an initiator wishes to respond to a message sent by the target it indicates this by asserting $\overline{ATN}$, before releasing $\overline{ACK}$ to finish the transfer. The target should then enter the Message In phase, and transfer message bytes until $\overline{ATN}$ goes inactive, up to 256 bytes. If the initiator attempts to send more than this the target will send the MESSAGE REJECT message and terminate the command with status of CHECK CONDITION and sense set to ABORTED COMMAND. If a parity error occurs during the message transfer, the target must wait until the transfer is complete, then instruct the initiator to resend all previous message bytes, by asserting $\overline{REQ}$ before changing phase. If the parity error occurs again the command will be terminated with status of CHECK CONDITION and sense ABORTED COMMAND.

The relevant functions for message phases are *messin( )*, *getmes( )* and *messout( )*. *messin( )* is used to send a message to the initiator, while *getmes( )* fetches a message from the initiator. *messout( )* processes the message sent by the initiator. The supported messages are now explained.

IDENTIFY: establishes the use of a greater message set, indicates the ability to support, or not support, disconnection and gives a LUN number, if necessary.

ABORT: if the reserving initiator sends this message to the target it causes the board to be reset.

BUS DEVICE RESET: This is similar to ABORT, except any initiator can implement it, resetting the board.

MESSAGE PARITY ERROR: On receiving this message the target attempts to resend the last message sent, and if this fails, terminates the command with status of CHECK CONDITION and sense set to HARDWARE ERROR.

MESSAGE REJECT: the targets response to this message is determined by the message being rejected. If the last message sent was COMMAND COMPLETE, the MESSAGE REJECT is ignored, since the initiator must support the mandatory message. A MESSAGE REJECT in reply to a DISCONNECT causes the disconnection to be cancelled. If it was in reply to a MESSAGE REJECT sent, the command is terminated with a status of CHECK CONDITION and sense set to HARDWARE ERROR. If the message sent was IDENTIFY (during reconnection) the target immediately goes to the Bus Free Phase and aborts the command. No Status or Message In phases are attempted, though sense is set to HARDWARE ERROR.

4.0 Run-Time Software (Continued)

It should be noted that *messin( )* has been written to only allow *getmes( )* to be called, from inside *messin( )*, once. Alternatively on parity error the target and initiator could eternally cycle, sending each other the MESSAGE PARITY ERROR message.

4.7.3 Non-Recoverable Errors

The following are errors which occur **during selection**, so severe that system operation is terminated, with an error code displayed on the LED.

The first two concern a DP5380.

Error 4
Wrong interrupt flags active.

Error 5
SEL inactive.

These errors concern either a DP5380 or a DP8490.

Error 6
SCSIID bit not active on bus.

Error 7
More than two bits active on bus.

The final errors are for a DP8490 only.

Error 8
SCSI parity error.

Error 9
Select flag inactive.

5.0 User Guide

The SCSI Printer Controller (SPC) is a Small Computer System Interface target board, which can use either the DP5380 Asynchronous SCSI interface (ASI) or DP8490 Enhanced Asynchronous SCSI Interface (EASI). It can be selected and used by an initiator as described in the ANSX3.131-1986 SCSI standard as defined by the ANSI X3T9.2 committee. This document will explain the installation of SPC, and show how it can be used in a system. By way of example a description is given of how the SPC can be used with an Advanced Storage Concepts ASC-88 IBM PC-SCSI Manager II™ host adaptor.

5.1 INSTALLATION

This section explains how the board must be set up before use, and the type of connectors required to interface to it. Any reference made to an EASI also applies for an ASI.

5.1.1 POWER

The SPC can be installed in a Personal Computer (PC) where it takes power from the backplane. There are two connections to +5V and two to ground. These are the only connections made to the backplane. Alternatively power can be taken from the available connector block *(Figure 5.1)*.

When the board receives power the LED will come on, and stay on if the board passes its self diagnostics. If it fails an error message will be displayed by the LED, indicating the source of error. This will be further explained in section 5.1.5. If the LED does not come on some fatal error has occurred.

5.1.2 SCSI CONNECTOR

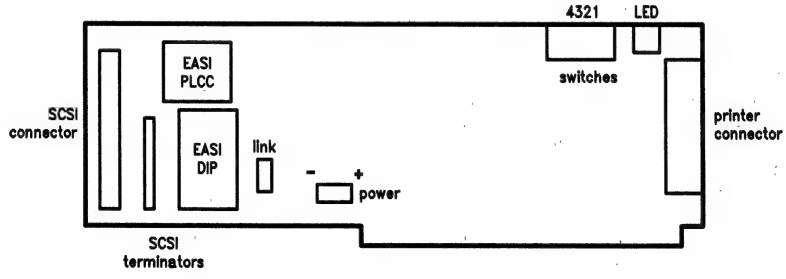
The SCSI bus should consist of a 50 way flat ribbon cable a maximum of 6.0 meters long. Both ends of this cable should have all SCSI lines terminated, with a 330Ω resistor to ground and a 220Ω resistor to power. SCSI devices are daisy chained along this cable, with SCSI signals common to all devices.

The SPC contains sockets for two DP8490 or DP5380 devices, one a PLCC socket the other DIL. **Only one of these sockets should contain a device.**

To comply with the regulations on terminating resistors six SIL resistor packs are available between the SCSI connector and the EASI DIL package *(Figure 5.1)*. If this board is not to be used at the end of the SCSI cable all of these six packs must be removed. The resistors have been socketed for this purpose.

An option available in the SCSI standard is to supply terminator power on the cable, so terminators at either end of the bus can use the same power. If the device at one end of the bus is unpowered its terminators can receive power from the cable, and the bus will operate correctly. It should be noted that other manufacturer's CMOS devices will not work in this configuration since if not powered they pull the SCSI lines low. National Semiconductor's DP5380 and DP8490 have special input protection to prevent this.

Pin 26 on the SCSI connector is the terminator power pin, which can be left floating by leaving the jumper in position 2, see *Figure 5.2*. By moving the jumper to position 1 the terminator power is supplied to this pin. *Figure 5.3* shows the two possible configurations. Terminator power is fed through a Schottky barrier diode to prevent a backflow of power into the board.



TL/F/10082–6

FIGURE 5.1

5.0 User Guide (Continued)

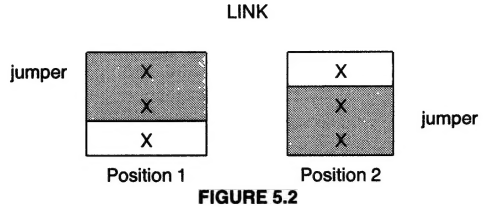
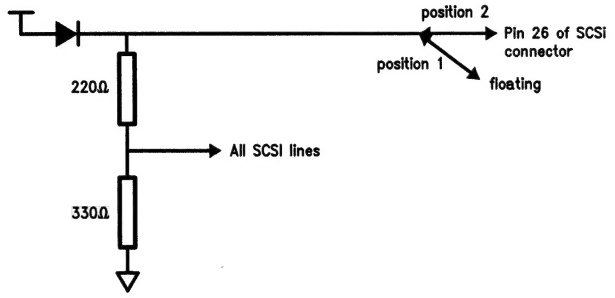


FIGURE 5.2



TL/F/10082–7

FIGURE 5.3

5.1.3 SWITCH BLOCK SETTINGS

The switch block is included to allow the user to select the SCSI ID of the SPC. This is used to identify the board during selection phases and determine its priority in arbitration. The SCSI ID is read in as a three bit binary number and converted in software to an eight bit pattern, with one bit active. For an ID of 0 the least significant bit is active, for an ID of 7 the most significant bit is active.

The three bit number is taken from the switch block (see *Figure 5.1*), using switches 1, 2 and 3. 1 is the most significant bit, 4 is unused. These switches should be set to give a unique ID for this bus. An ID of 0 is suggested, making the printer the lowest priority device on the bus. These switches must be set up before power is applied, as they are only checked during the board diagnostics.

5.1.4 PRINTER CONTROLLER

The printer connector is a standard IBM 25 way 'D' type connector. The SPC software controls data transfers to the printer using the BUSY and $\overline{\text{STROBE}}$ signals.

5.1.5 ERROR MESSAGES

Non-recoverable errors cause a four bit binary number to be displayed by the LED. This indicates the source of error.

The LED displays the number by occulting for 1 second to show a 1, ½ second to show a 0. The code is displayed most significant bit first, with the LED on for 1 second between bits. The error number is repeatedly displayed, between breaks of 2 seconds when the LED is on.

The error numbers, and their cause, are shown below. The first four errors are generated during the board diagnostics, the others occur when a selection fails. The SCSI controller will be referred to as an EASI, unless the error is specific to either a DP5380 or DP8490. These possible differences are due to the DP8490's more extensive testing, and because it handles a selection differently.

Error 0

The DMA could not be accessed. This is possibly a damaged device.

Error 1

EASI can not be accessed. Device could be damaged, or not properly terminated.

Error 2

DP8490 has failed loopback test. Device could be damaged.

Error 3

DP8490 has failed the loopback DMA test. In this test a DMA transfer to memory from the DP8490 is attempted. Failure here could indicate an error in memory, processor, DMA, DP8490 or PALs.

These first four errors concern problems internal to the board. The following errors indicate a bus problem that occurred while the board was waiting for selection.

Error 4

DP5380 error flag active.

Error 5

DP5380 found select line inactive. Possible selection timeout or bus error.

Error 6

EASI ID bit not active on bus. Possible selection timeout or bus error.

Error 7

EASI detected more than two bits active on bus. During a selection phase the initiator can only assert its own ID and the targets ID on the bus. This indicates a bus error.

Error 8

DP8490 SCSI parity error. Error on bus.

Error 9

DP8490 select flag inactive. This is board hardware error, possibly in DP8490 or PAL, possibly interrupt line.

For any of these problems the user should try switching the device on and off again. For bus errors the cable and terminators should be checked along with any other devices on the bus.

5.0 User's Guide (Continued)

5.2 DRIVER SOFTWARE

By way of example it will be shown how software can be written to drive the SPC from an ASC-88 host adaptor, installed in a PC. The ASC-88, like many commercially available host adaptors, handles all low level SCSI signal controls. The user controls the command to be implemented by means of a Job Control Block, JCB, which is passed to it.

The software required to use an ASC-88 is on the supplied floppy disk. This disk contains both the source code, explained later, and the executable code. This is called ***printout.exe*** and can be implemented in the form:

printout *filename*

The *filename* supports the MS-DOS use of directories and paths:

e.g., printout a: /ASC /ASC.C

The second executable command, called ***stoprint.exe***, can be used to terminate a print. When this command is executed the SPC flushes the print queue and initializes the printer.

5.2..1 SEQUENCE OF COMMANDS

Although the SPC requires no prescribed sequence of SCSI commands **an initiator must reserve the unit before a print can take place**. Otherwise the sequence of commands specified here indicate how the SPC could be used.

Any print should begin with a TEST UNIT READY command. On receiving this command the SPC tests the printer to ensure it has paper and is not in an error condition. If the status of GOOD is returned the user should then attempt RESERVE UNIT. If the command is successful the SPC will only execute commands from this initiator. This prevents print data from two or more sources being mixed.

The user then sends PRINT commands with a maximum length of data per transfer of 2 kB. If the file to be printed is larger than 2 kB the data must be sent in several blocks. This limit, set to minimize bus latency, is more fully explained in Section 4.4.3.

These PRINT commands transfer the data to the SPC print buffer. To check if an error occurred when the SPC attempted to transfer the data to the printer the user can send a REQUEST SENSE command. If the returned data is NO SENSE the printer is still operational. Any other sense indicates an error.

The final command sent is RELEASE UNIT. This allows the SPC to be used by other initiators.

5.2.2 ASC-88 SOFTWARE

All ASC-88 software is written for a Microsoft® C compiler. File ASC.C contains the main run-time software. ASC STRUC.LIB sets up the structure which is used as a Job Control Block, while ASCCOM.LIB defines the JCBs for particular SCSI commands. The routines used in ASC.C, other than SCSI command calls, are in ASCROT.LIB and UTIL.LIB. CONSTANT.H contains the constants used throughout these files. If the user wishes the SCSI ID of the target board to be anything other than zero they should change the value of *TARGET_ID* in this file, and recompile the code for both commands.

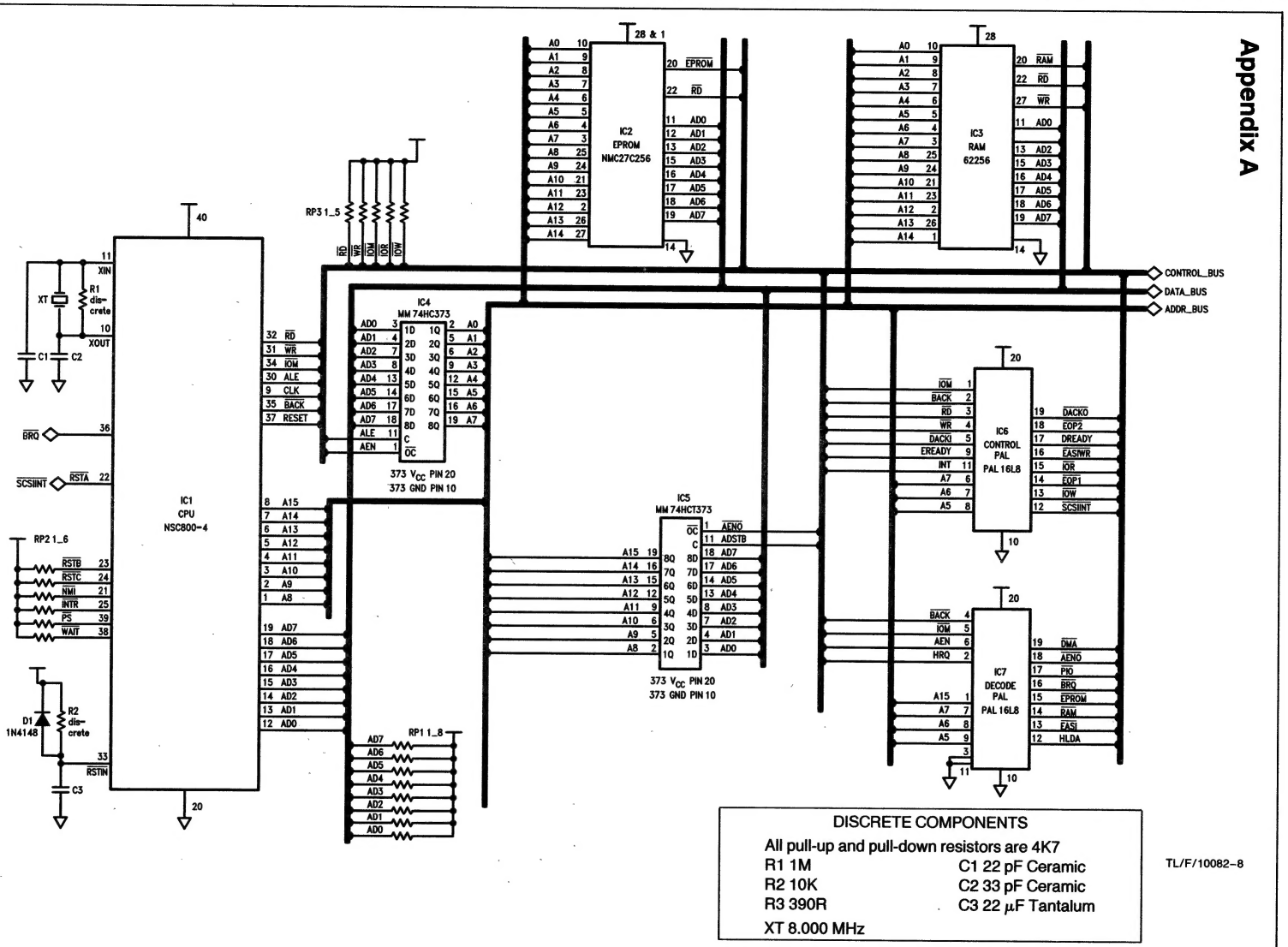
STDIO.H is a Microsoft library containing constants, macro definitions and function declarations for I/O stream operations. This controls opening files, reading files, detecting file ends and closing files. DOS.H, also a Microsoft library, handles the interface to MS-DOS®. This allows the user to set up registers and execute interrupts. The final Microsoft library CONIO.H allows the user to fetch information from the keyboard.

The SCSI-BIOS™ EPROM in the ASC-88 is accessed by generating interrupt number forty. The value in the 'AH' register determines what SCSI-BIOS software is used. SCSI-PRO™ implements the SCSI command specified in the JCB whose starting address is passed in the 'BX' and 'ES' registers.

ASC.C must be compiled and loaded to produce a file printout.exe. When executed this code opens the file specified in the command line and instigates the sequence of commands, outlined in the previous section, to print it out. After these commands the file is closed.

If the printer enters an error state while communicating with the initiator a message will be displayed on screen and the user is given the option of terminating the print. If the print is to be continued the user must correct the printer error and press a key to restart the print. If an error occurs when the SPC has already received all the data from the PC the board will wait until the print error is corrected and continue printing.

STOPRINT.C contains the run-time software required to terminate a print. This uses the library CHECK.LIB, which checks the success of a FLUSH BUFFER command. To terminate an unwanted long print the user can: take the printer off-line; press escape to leave *printout*; send the *stoprint* command.

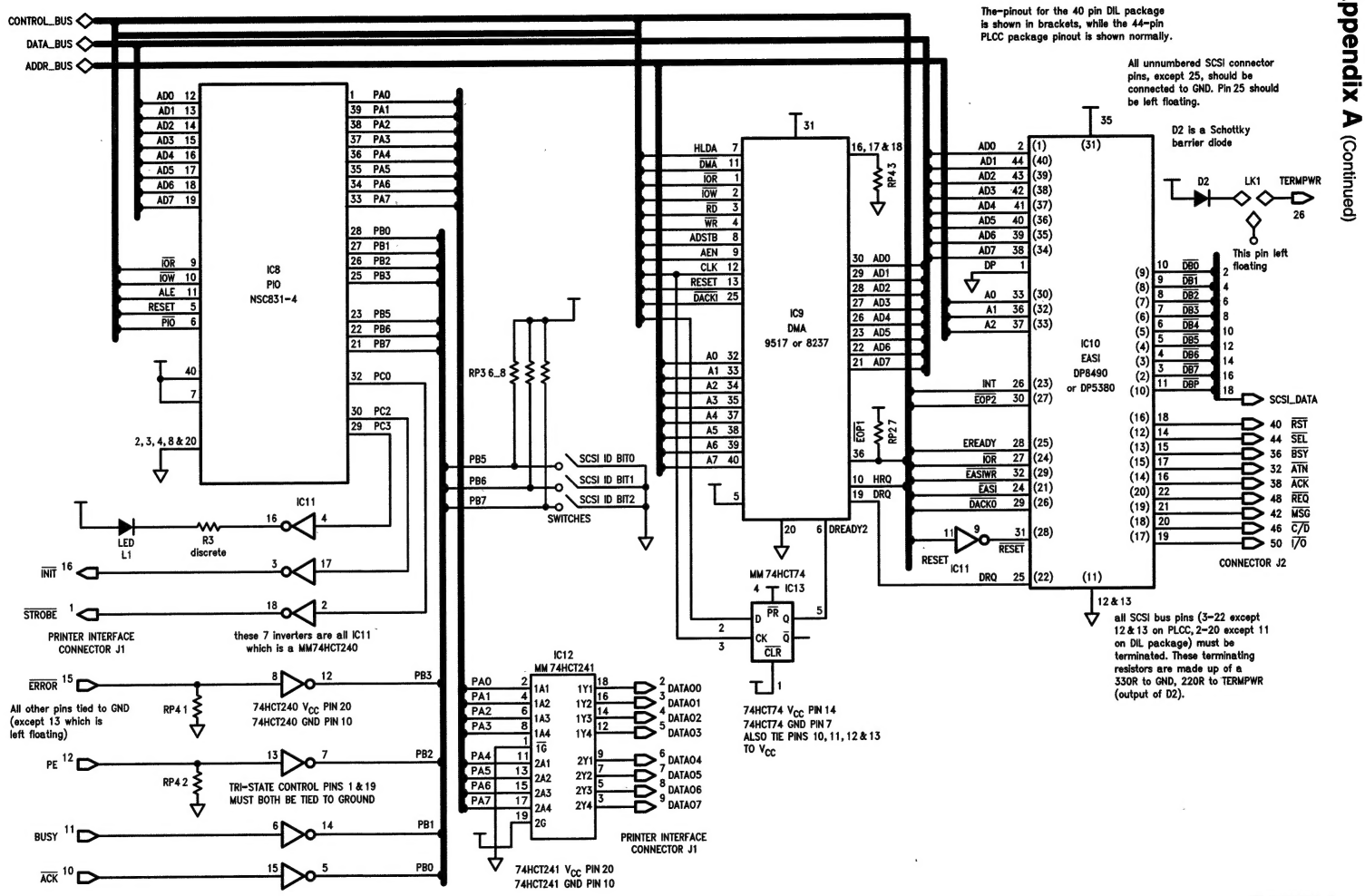


DISCRETE COMPONENTS

All pull-up and pull-down resistors are 4K7

R1 1M	C1 22 pF Ceramic
R2 10K	C2 33 pF Ceramic
R3 390R	C3 22 μF Tantalum
XT 8.000 MHz	

TL/F/10082–8


CONTROL_BUS
DATA_BUS
ADDR_BUS
IC8
PIO
NSC831-4
IC9
DMA
9517 or 8237
IC10
EASI
DP8490
or DP5380
IC11
IC12
MM 74HCT241
IC13
MM 74HCT74
The pinout for the 40 pin DIL package is shown in brackets, while the 44-pin PLCC package pinout is shown normally.
All unnumbered SCSI connector pins, except 25, should be connected to GND. Pin 25 should be left floating.
D2 is a Schottky barrier diode
This pin left floating
SCSI_DATA
CONNECTOR J2
SCSI ID BIT0
SCSI ID BIT1
SCSI ID BIT2
SWITCHES
LED
L1
R3
discrete
PRINTER INTERFACE
CONNECTOR J1
these 7 inverters are all IC11 which is a MM74HCT240
74HCT240 VCC PIN 20
74HCT240 GND PIN 10
All other pins tied to GND (except 13 which is left floating)
TRI-STATE CONTROL PINS 1 & 19 MUST BOTH BE TIED TO GROUND
74HCT241 VCC PIN 20
74HCT241 GND PIN 10
74HCT74 VCC PIN 14
74HCT74 GND PIN 7
ALSO TIE PINS 10, 11, 12 & 13 TO VCC
all SCSI bus pins (3–22 except 12 & 13 on PLCC, 2–20 except 11 on DIL package) must be terminated. These terminating resistors are made up of a 330R to GND, 220R to TERMPWR (output of D2).

TL/F/10082–9